TESIS DE LA UNIVERSIDAD DE ZARAGOZA
2025
264

Sofia Hustiu

Temporal logic objectives in high-level path planning: discrete-event-based conceptual framework

Director/es
Cezar Pastravanu, Octavian
Mahulea Poleuca, Cristian Florentin

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

Universidad de Zaragoza
Servicio de Publicaciones

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

TEMPORAL LOGIC OBJECTIVES IN HIGH-LEVEL PATH PLANNING: DISCRETE-EVENT-BASED CONCEPTUAL FRAMEWORK

Autor

Sofia Hustiu

Director/es

Cezar Pastravanu, Octavian
Mahulea Poleuca, Cristian Florentin

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

Programa de Doctorado en Ingeniería de Sistemas e Informática

2025

Universidad Zaragoza
1542

Tesis Doctoral

TEMPORAL LOGIC OBJECTIVES IN HIGH-LEVEL PATH PLANNING: DISCRETE-EVENT-BASED CONCEPTUAL FRAMEWORK

Autor

Sofia HUȘTIU

Director/es

Prof. Dr. Eng. Cristian Florentin MAHULEA (University of Zaragoza)
Prof. Dr. Eng. Octavian-Cezar PĂSTRĂVANU („Gheorghe Asachi" Technical University of Iași)

Universidad de Zaragoza
Escuela de Doctorando
Programa de Doctorando en Ingeniería de Sistemas e Informática
2025

I would like to dedicate this thesis to my loving parents . . .

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Prof. Pastravanu Octavian-Cezar (Technical University of Iasi, TUIASI) and Prof. Mahulea Cristian (University of Zaragoza, Unizar, Spain), for their patience, support, incredible guidance and lot of encouragement throughout the entirety of my PhD journey. Their mentorship has been essential, both academically and personally. I am equally indebted to Prof. Kloetzer Marius (TUIASI), with whom I worked closely. His insightful feedback, patience, and dedication have greatly elevated my work and motivated me to grow as a researcher and as a person.

Secondly, I am grateful for the collaboration and support of Prof. Dimarogonas Dimos (KTH Stockholm, Sweden), who provided valuable expertise and guidance throughout my three-month research stay during my PhD studies. My sincere appreciation also go to Prof. Ezpeleta Joaquin (Unizar), whose support extended further than my time in Zaragoza and whose constructive discussions are still crucial in modeling my research. Additionally, I would like to thank Prof. Jean-Jacques Lessage (Université ENS Paris-Saclay, LURPA, Gif-sur-Yvette, France) for his valuable feedback and insights during the preparation of a journal article, which later led to its successful acceptance.

Thirdly, I would like to express my gratitude to Prof. Caruntu Constantin and Prof. Alexandrescu Adrian for their encouragement and guidance, and to my PhD committee members, Prof. Burlacu Adrian and Prof. Matcovschi Mihaela, for their constructive feedback and valuable suggestions towards my research. I would like to address special thanks to Prof. Eduardo Montijano (Unizar) for his support and fruitful discussions. Many thanks also to Jose Merseguer (Unizar), Tord Christer Magnusson (KTH), and Elena Constantin (TUIASI) for their administrative support, in ensuring the smooth process of my doctoral studies.

I am deeply grateful to my mentors at Continental Romania SRL, specifically to Lacatusu, Vladut, and Iulian, for their encouragement during my PhD studies, and to my dear colleagues Roxana and Daniel for their support and comradery.

To my dear colleagues at Unizar, I would like to thank you for your support, friendship, and the beautiful experiences that we have shared. Alejandro, Bruno, Cesar, Edgar, Eduardo, Eva, Fanny, Irene, Jesus, Laura (my roommate, from whom I learned a lot about academia and life), Lucia, Nacho, Rafael, Rodrigo, Sebastian, and Tania, all of you have made my

nine-month stay in Zaragoza special. Your friendship has been supported not only through words, but also by sharing delicious Spanish and Mexican food, from finding comfort in our common challenges, turning sorrows into laughter over drinks, and by staying positive and enjoying every moment of our time together. The friendship bond that we have built continues to grow, as our mutual support is observed through periodical discussions that we share in our daily lives.

Similarly, I am grateful to my colleagues at KTH, who taught me in the short period of three-month research stay, the value of collaboration in a truly international research environment. Dear companions: Adam, Amritam, Elis, Erfaun, Esteban, Joana, Maria, Mayank, Miguel, Mina, Nana, Pedro, Rijad, Robert, and Xiao thank you for showing me the value of productive research discussions, for your tolerance of helping each other, and for including me in your wonderful group. The time we spent together has been a memorable experience, full of learning and growth in a place that I consider an international research hub.

My sincere appreciation goes also to my colleagues and friends at TUIASI: Alex, Alexa, Andrei, Aura, Codrin, Delia, Flavius, George, Ionut, Iuliana, Otilia, Paul, Razvan, Stefan, and Tudor. All of you have become much more than colleagues: you became trusted companions with whom I hope to have a lifelong connection. The bond that we share helped me throughout the challenges of my PhD, and your kind words gave me strength in times of need. I am grateful for the beautiful memories that we have created together, for the daily challenging puzzles that we have finished so far, for our gastronomic evenings and research nights, and for your encouragement and patience.

A special gratitude goes to my dear friends Alina and Cristina, who have supported and encouraged me since before my PhD studies. Both then and now, their close friendship has been a source of strength that I continue to grow in my academic career.

Lastly, I want to express my deepest gratitude to my family: both of my parents, my grandmother, and especially my sister, whose unconditional love, patience, and belief in me have been the foundation of my professional and personal success. I dedicate this achievement to them.

Thank you all for being part of this incredible journey. This accomplishment would not have been possible without each and every one of you.

Abstract

This thesis proposes novel Discrete Event System (DES)-based frameworks under Petri net formalism that provide planning solutions for homogeneous and heterogeneous multi-robot systems and ensure high-level missions. These missions are commonly described using formal specifications such as Linear Temporal Logic (LTL), which capture complex temporal and spatial requirements. Alternative high-level formalisms, such as Metric Interval Temporal Logic (MITL), extend the LTL capabilities by incorporating time intervals.

The first contribution of this thesis is a task decomposition method that reduces the complexity of LTL global missions by dividing them into smaller, independent tasks.

Secondly, a planning strategy aims to compute trajectories for a robotic team satisfying a temporal logic mission, by providing a global view of the robotic team's state. The planning solutions are obtained through both dynamical programming and model-checking approaches. Specifically, two representations are joined into a single Petri net denoted *Composed Petri net*, considering the robotic, respectively mission models united through an intermediate layer of places associated with a set of regions of interest that the robots should reach and/or avoid according to the high-level specification. The versatility of the proposed joined model captures both spatial and temporal constraints of the multi-robot system concerning the workspace, considering LTL and MITL specifications. The latter specification is incorporated into a model denoted *Composed Time Petri net*.

The third contribution enhances the planning method by enabling parallel robot movement based on precomputed mission trajectories. It considers the free space as shared resources and applies the Banker's algorithm to prevent deadlocks. This approach improves efficiency, ensures coordinated, collision-free motion, and fulfills global missions even in resource-constrained scenarios like narrow passages.

Finally, a motion planning solution addressing the coordination of a heterogeneous robotic system ensuring an LTL mission is developed on the Nets-within-Nets (NwN) paradigm. The newly proposed framework, namely the *High-Level robotic team Petri net*, examines a hierarchical Petri net structure capturing both local robot behaviors and global mission constraints. Furthermore, a proposed synchronization function enables the movement of robots with different capabilities. Hence, this solution enables an easier grip on the multi-robot systems where heterogeneity can introduce additional complexity, and conventional approaches may struggle to ensure coordination and scalability.

Rezumat

Această teză propune cadre noi bazate pe Sistemele de Evenimente Discrete (SED), utilizând formalismul rețelelor Petri, care oferă soluții de planificare pentru sisteme multi-robot omogene și eterogene și asigură îndeplinirea unor misiuni de nivel înalt. Aceste misiuni sunt, de obicei, descrise utilizând specificații formale precum Logica Temporală Liniară (LTL), care captează eficient cerințe complexe temporale și spațiale. Formalisme alternative de nivel înalt, precum Logica Temporală Metrică pe Interval (MITL), extind capacitățile LTL prin integrarea intervalelor de timp explicite.

Prima contribuție a acestei teze este reprezentată de o metodă de decompoziție a sarcinilor care reduce complexitatea misiunilor globale LTL prin împărțirea acestora în sarcini mai mici și independente.

În al doilea rând, o strategie de planificare are ca scop calcularea traiectoriilor pentru o echipă robotică, satisfăcând o misiune descrisă prin logică temporală, oferind o viziune globală asupra stării echipei robotice. Soluțiile de planificare sunt obținute prin abordări bazate pe programare dinamică și verificarea modelelor. Mai exact, două reprezentări sunt unite într-o singură rețea Petri, denumită *Rețea Petri Compusă*, luând în considerare modelele robotice și de misiune, unite printr-un strat intermediar de locuri asociate cu un set de regiuni de interes pe care roboții trebuie să le atingă și/sau să le evite, conform specificațiilor de nivel înalt. Versatilitatea modelului compozit propus captează atât constrângerile spațiale, cât și cele temporale ale sistemului multi-robot în raport cu spațiul de lucru, luând în considerare specificațiile LTL și MITL. Specificația MITL este încorporată într-un model denumit *Rețea Petri Compusă cu Timp*.

A treia contribuție îmbunătățește metoda de planificare prin permiterea mișcării paralele a roboților pe baza traiectoriilor de misiune precompute. Se consideră spațiul liber ca resursă partajată și se aplică algoritmul Bancherului pentru a preveni blocajele. Această abordare îmbunătățește eficiența, asigură o mișcare coordonată, fără coliziuni, și îndeplinește misiunile globale chiar și în scenarii cu resurse limitate, cum ar fi pasaje înguste.

În cele din urmă, o soluție de planificare a mișcării care abordează coordonarea unui sistem robotic eterogen și asigură o misiune LTL este dezvoltată pe baza paradigmei*Nets-within-Nets (NwN)*. Noul cadru propus, denumit *Rețea Petri de Nivel Înalt pentru o echipă robotică*, examinează o structură ierarhică a rețelelor Petri, care captează atât comportamentele locale ale roboților, cât și constrângerile globale ale misiunii. În plus, o funcție

de sincronizare propusă permite mișcarea roboților cu capacități diferite. Astfel, această soluție facilitează gestionarea sistemelor multi-robot, unde eterogenitatea poate introduce o complexitate suplimentară, iar abordările convenționale pot avea dificultăți în asigurarea coordonării și scalabilității.

Resumen

Esta tesis propone nuevos marcos basados en Sistemas de Eventos Discretos (SED) bajo el formalismo de redes de Petri, que ofrecen soluciones de planificación para sistemas multi-robot homogéneos y heterogéneos y garantizan misiones de alto nivel. Estas misiones suelen describirse utilizando especificaciones formales como la Lógica Temporal Lineal (LTL), que captura de manera efectiva requisitos temporales y espaciales complejos. Formalismos alternativos de alto nivel, como la Lógica Temporal de Intervalos Métricos (MITL), amplían las capacidades de la LTL al incorporar intervalos de tiempo explícitos.

La primera contribución de esta tesis está representada por un método de descomposición de tareas que reduce la complejidad de las misiones globales descritas en LTL al dividirlas en tareas más pequeñas e independientes.

En segundo lugar, se presenta una estrategia de planificación que tiene como objetivo calcular trayectorias para un equipo robótico que satisfagan una misión descrita en lógica temporal, proporcionando una visión global del estado del equipo robótico. Las soluciones de planificación se obtienen mediante enfoques de programación dinámica y verificación de modelos. Específicamente, se combinan dos representaciones en una única red de Petri denominada *Red de Petri Compuesta*, que considera los modelos robóticos y de misión unidos a través de una capa intermedia de lugares asociados a un conjunto de regiones de interés que los robots deben alcanzar y/o evitar según las especificaciones de alto nivel. La versatilidad del modelo combinado propuesto captura tanto las restricciones espaciales como las temporales del sistema multi-robot en relación con el espacio de trabajo, considerando especificaciones LTL y MITL. La especificación MITL se incorpora en un modelo denominado *Red de Petri Compuesta con Tiempo*.

La tercera contribución mejora el método de planificación al permitir el movimiento paralelo de los robots basado en trayectorias de misión precomputadas. Considera el espacio libre como un recurso compartido y aplica el algoritmo del Banquero para prevenir bloqueos. Este enfoque mejora la eficiencia, garantiza un movimiento coordinado y libre de colisiones, y cumple con las misiones globales incluso en escenarios con recursos limitados, como pasajes estrechos.

Finalmente, se desarrolla una solución de planificación de movimiento que aborda la coordinación de un sistema robótico heterogéneo, garantizando una misión LTL basada en el paradigma de *Nets-within-Nets (NwN)*. El nuevo marco propuesto, denominado *Red de Petri*

de Equipo Robótico de Alto Nivel, examina una estructura jerárquica de redes de Petri que captura tanto los comportamientos locales de los robots como las restricciones globales de la misión. Además, se propone una función de sincronización que habilita el movimiento de robots con diferentes capacidades. Por lo tanto, esta solución facilita el manejo de sistemas multi-robot donde la heterogeneidad puede introducir una complejidad adicional, y los enfoques convencionales pueden tener dificultades para garantizar la coordinación y la escalabilidad.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The evolution of path planning in robotics began with the fundamental problem of guiding a robot from a starting point to a final destination while avoiding obstacles. This known issue, known as classical *navigation problem*, has significantly advanced over the years, especially with the introduction of multi-robot systems (also known as a team of robots). As robotics systems became more sophisticated, motion planning evolved to address more complex scenarios, such as coordinating the movements of multiple robots in static and/or dynamic environments. Since a set of robots represents a robotic system, their routes are designed in terms of a single mission that should be accomplished, including synchronization and sequencing regarding the set of tasks that should be ensured. The solution to this problem is commonly established as *path planning*. Through task, it is understood as an objective (such as an action, e.g., visiting a region of interest, picking a package) that at least one robot should fulfill autonomously, considering the workspace that the robots evolve. Throughout the thesis, most tasks are represented by the visit and/or avoidance of regions of interest included in the workspace.

In multi-robot systems, motion planning must consider the physical obstacles and the interactions between robots. This is particularly critical in scenarios where the robots may need to navigate either in known or unknown environments, such as warehouses. For example, the robots must efficiently plan an optimal path based on prior knowledge of the space, while also adapting to new obstacles and avoiding them, such as other robots or the workers operating in the same space. In industrial scenarios, teams of robots may work collaboratively to map unknown spaces, coordinating their efforts to ensure full coverage while avoiding any collision that might occur. The ability to plan collision-free paths for multiple robots, even in unpredictable workspaces, has become critical.

1.1 Context

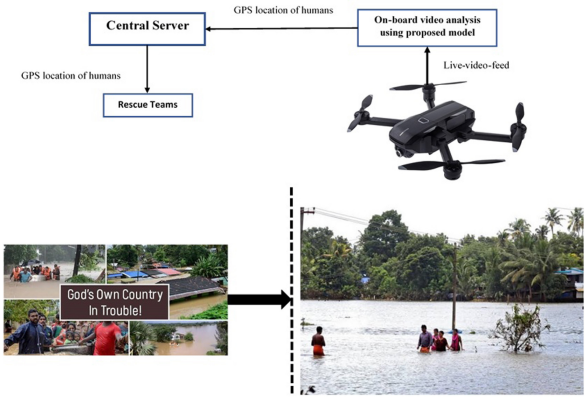
The importance of planning trajectories and task assignment for multi-robotic systems lies in its ability to optimize performance and ensure successful mission completion in various complex applications, such as search and rescue operations [5], autonomous transportation [6], and industrial manufacturing [7]. Both 2D and 3D workspaces benefit from various planning solutions improving the tasks that should be ensured, considering teams of mobile robots, respectively Unmanned Aerial Vehicles (UAVs), also known as drones. For example, in the agriculture domain, the relevance of UAVs includes domains such as using spray systems [8], crop data acquisition, and examination [9]. Besides these applications among others [10], several activities can be improved based on automated UAVs, having a beneficial impact on human safety.

For easier visualization of the applications that strengthen the use of autonomous robots, Figure 1.1 includes several illustrative examples, such as: (a) monitoring an indoor environment by using an autonomous vacuum cleaner robot which includes several sensors for humidity, gas, and temperature, (b) drone surveillance in search and rescue problems, by scanning and identifying the location of the people that should be rescued, (c) autonomous package delivery in order to prevent the spread of the COVID-19 virus, and (d) acquiring images of the tomatoes for training purposes leading to automatic handling of the fruits.

Motion planning for mobile robots initially focused on single-robot trajectory computation, as detailed in [15], and gradually extended to multi-robot systems. Throughout this thesis, the *robot* shall also be referred also as an *agent* to highlight that the presented methods and solution are not restricted to particular types of robots such as mobile robots, industrial robots, or drones. Effective coordination of agents can significantly enhance efficiency, reduce mission time, and minimize human intervention, which is particularly crucial in time-sensitive and hazardous environments [16]. The planning methods in the literature aim to ensure that a given mission, often expressed through high-level formal specifications, is accomplished efficiently and safely. Formal methods play a significant role in multi-robot trajectory planning, specifying global missions that the team must collectively achieve [17].

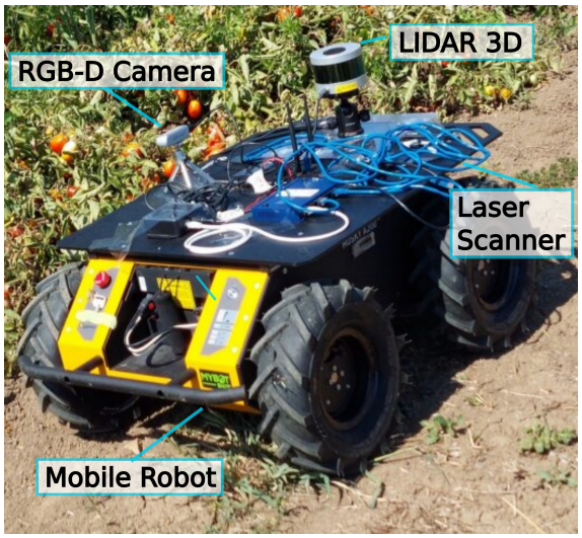
One of the simplest missions formally described could include specifications requiring the visit and/or avoidance of a set of regions of interest for the team of robots. In other words, the mission states that a set of regions should be reached by the robotic team, e.g., for picking up a package while avoiding the obstacles present in the working space. An intuitive approach to expressing this type of mission is presented in [18], considering the Boolean operators. For example, a mission formulated as ”visit region A and avoid region B” could be formally defined as a Boolean expression using conjunctions and negations. Once the mission progresses towards a more complex scenario, involving sequencing or parallelism, the Boolean operators are not rich enough to encode such specifications, e.g., ”visit region A, then region C, while always avoiding region B”. One structured formalism

(a) Autonomous vacuum cleaner [11]



(b) UAV for search and rescue [12]

(c) Autonomous package delivery [13]



(d) Mobile robot for agriculture [14]

Fig. 1.1. Various applications for autonomous robots

suitable to express these missions is known under the name Linear Temporal Logic (LTL), in which Boolean operators are combined with temporal ones [19, 4]. This formalism is widely used in literature due to its advantages [20] in checking the compatibility through model checking tools [21], intuitive expression of missions, and easy-to-handle representation such as Büchi automaton that benefit of mature algorithms. Several works have adapted these high-level specifications for both local (addressing individual missions to each agent) and global (addressing specification for the entire team) missions within multi-robot systems [22, 23].

The LTL formalism is a complex language that allows various fragments to be part of, such as Generalized Reactivity(1) (GR(1)) [24] and co-safe LTL [25]. Moreover, the LTL itself represents a fragment from full Computation Tree Logic (CTL*), a language that contains not only the Boolean and temporal operators but also path quantifiers, which are part of the CTL formalism (also a fragment from CTL*) [26, 27]. The difference between these two formalisms is that LTL defines a single timeline in which the mission could be fulfilled, while CTL considers multiple potential futures, i.e., ”when region A is reached, then all future paths should reach region C”. As stated in [27], the CTL* language is used in the development and checking of the correctness of complex systems, since an automata can be associated with the formula.

Despite the stated benefits, several challenges are brought by the previously mentioned formalisms, i.e., LTL [28], from which it can be enumerated the absence of time constraints, e.g., ”visiting region A in 5-time units” or expressing uncertainty, e.g., ”the probability of visiting region B within 5-time units is at least 95%”. Considering the fact that various specifications can be applied to different planning scenarios, the researchers have explored a variety of specification languages allowing for complex missions with both spatial and temporal dependencies. Some formalisms convey the mission by the use of Boolean predicates evaluated through discrete or continuous time: (i) Metric Temporal Logic (MTL) expressing explicit time intervals, e.g., ”reach region A in exact 3-time units” [29], (ii) Metric Interval Temporal Logic (MITL) considering permissive time intervals, e.g., ”eventually visit region A within the next three-time units while always avoiding region B” [30], (iii) Probabilistic Computation Tree Logic (PCTL), e.g., ”the probability of reaching region A is at least 90%” [31], (iv) Time Window Temporal Logic (TWTL), e.g., ”stay in region A for 3-time units within the time frame [0,5]” [32]. Other formalisms tackle the continuous time with predicates over real-value, such as signal Temporal Logic (STL), e.g., ”always in the time interval [0,4], the error on the x-position of the robot shall be under 0.1[m]” [33], Continuous Stochastic Logic (CSL), e.g., ”the probability of avoiding obstacles in time interval [0,8] is at least 90%”.

Among the multitude of formalisms that are being defined in the literature, only a part of them are listed above, out of which some of them are used actively in planning trajectories for the robots: LTL, MTL, MITL, STL, and TWTL. One example can be inspected in [34],

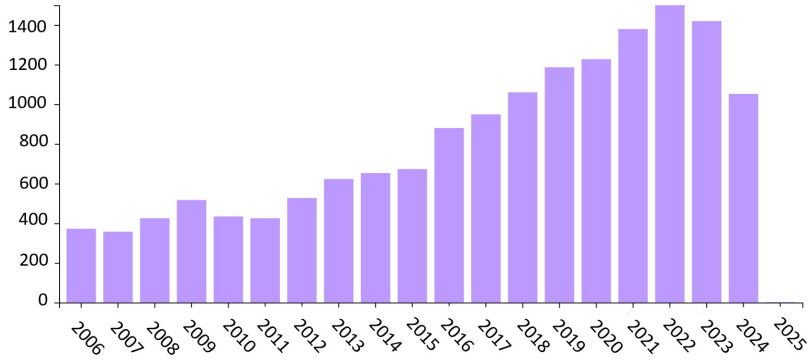
where the vehicle routing problem is solved by a multi-UAV team ensuring an optimal path for MTL specifications. The planning strategies might differ, depending on the complexity of the problem. For instance, in centralized approaches a single *global mission* is provided to the entire robotic system, e.g., in the form of LTL [35], where the agents should work together to ensure the totality of the spatial and temporal constraints. The distributed approach relies on imposing individual tasks on each robot while having access to local information only e.g., tasks under TWTL [36], LTL, or STL [37] formalisms. The last enumerated strategy is based on the fact that the agents share information if they are near each other, useful in tasks requiring coordination.

Recent works, such as [38], propose integrating high- and low-level path planning strategies to handle MITL tasks, and in [37] a solution incorporating both LTL and STL tasks is presented. Cooperation among team agents is also tackled in literature, considering either individual MITL missions [39], either individual LTL missions [40].

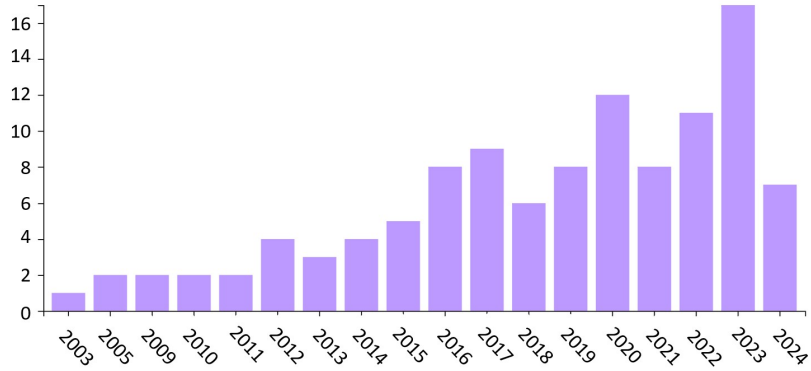
The relevance of temporal logic formalism in the motion planning robotic field is emphasized in Figure 1.2, portraying an increased trend of research papers while searching (a) ”path planning robots” and the combination between (b) ”path planning robots” and ”temporal logic”. Particularly, in the last 20 years, the number of works published on the platform Web of Science has grown, since there are still many problems to be solved in the field of multi-agent system motion planning, such as collision avoidance in dynamic and uncertain environments, coordination of heterogeneous robotic teams and handling complexity when the number of robots increases in the team [41, 42]. Note that Figure 1.2 (b) illustrates the number of papers containing the exact phrases ”path planning robots” and ”temporal logic”, without adding the total number of papers that explore a particular research idea expressed through other standard formulation in the title, e.g., motion planning under STL specification, high-level planning.

The complexity of a planning strategy involving a multi-robot system has two-folded reasons: *the richness of the given global mission* and *the robotic model*. Firstly, depending on the application, a complex global mission given for a multi-robot system can provide an efficient solution only if the tasks are decomposed and assigned independently to the robots. Numerous methods have been proposed for decomposing global LTL missions enabling scalable planning for large teams of robots, but most rely on simplifying assumptions or specific LTL sub-classes [23, 43, 44]. Since the simplification of the problem space is relevant in motion planning strategies involving multi-robotic systems, in this thesis, a decomposition algorithm is proposed resulting in a set of individual smaller tasks allocated to the robots, considering a global LTL mission. Thus, the state space is significantly reduced compared with the centralized approaches.

Secondly, the chosen robotic model that outputs the paths influences the complexity of the planning strategy, directed towards the increase of the agents in the robotic team. Diverse representations under the Discrete Event Systems (DES) concept such as Transition



(a) Publication years graphic for ”path planning robots”



(b) Publication years graphic for ”path planning robots” and ”temporal logic”

Fig. 1.2. Analyze search results based on Web of Science database

Systems (TS) and Petri nets (PN) are employed to represent robots' movements and facilitate trajectory planning in complex environments since these representations are graph-based approaches [1, 44–46]. Due to a clear mathematical understanding of this formalism, as presented here [47], the DES representations enable the investigation of several planning problems, such as industrial manufacturing. Several papers focused on modeling TS or PN to facilitate the evaluation of the formal frameworks [48, 49].

Let us now discuss the use of these two main representations. In the case of TS, the planning strategy usually considers the modeling of each agent through an automata, while the global state of the robotic system is visualized through a product automata resulting from the combination of each robot's dynamic concerning its working space. In this scenario, challenges such as the exponential growth of state spaces, often referred to as the state explosion problem, remain a central issue [50].

In contrast, most Petri net-based planning strategies use a single model for the entire team's movements, regardless of the number of robots but dependent on the environment [4]. Therefore, the Petri net-based approaches offer a stable topology independent of the number of robots, as seen in [51] and [52], where changes in the environment and task planning are

incorporated into the Petri net model. This representation could also lead to state explosion if the reachability state-space is generated and the solution is searched among all the states of the robotic team. To avoid this problem, most researchers use structural approaches based on mathematical programming methods leading to a path planning solution.

Other papers extend the use of the Petri net model for the robotic model by adopting different classes of nets to encode additional information about the robotic system and/or about the workspace. For example, in [53] a Colored Petri net representation is used for a flexible manufacturing system, focusing on the deadlock avoidance problem and exploiting the properties of colored tokens to differentiate between processes. In [54] the authors associate a Timed Petri net modeling the environment by adding time constraints regarding the moving time of a robot from a region of interest to an adjacent one. The planning method solves an optimization problem by minimizing the time the robots should reach the regions of interest. Another approach for the use of the Time Petri net is observed in [55], where clinical pathways are optimized, considering the motivation from the COVID-19 pandemic.

The time requirements representation for a robotic system are studied in [56], where a Timed Colored Petri net model has been explored since the Colored Petri net allows the use of methods that reduce the complexity of the robotic model, where the robots have different capabilities (heterogeneous team). Particularly, in this paper, the authors propose a planning strategy for a robotic team that should fulfill three types of tasks: common (ensured by any type of robot), exclusive (required to be ensured by a fixed type of robot), and collaborative (ensured by multiple types of robots). The planning solution is returned by an optimization problem which includes all these types of tasks.

The motion planning field considering the high-level planning, supports both the sequential and parallel execution of different models, considering a model for the robotic team, and another one for the given mission. Through sequential planning, it is understood that one of the models is handled first, e.g., the mission model, computing a solution that ensures the requirements of the mission, followed by the manipulation of the second model, e.g., the robotic one, such that the robotic movements ensures the solution returned by the mission model.

When the mission is formally given under a high-level specification that relies on a mathematical background, as presented previously, then the planning strategies benefit from the advantages of two types of models (robotic system and the mission). Let us consider the sequential planning method proposed in [4], where a Petri net model represents the robotic team, while the LTL mission is modeled by a Büchi automaton. The planning solution of the robotic team implies that the robots follow a generated run from the automaton. Therefore, the approach considers first the automaton of the mission for which a finite number of runs ensure the mission is computed. Secondly, the robots attempt to follow a run throughout the robotic Petri net model. The entire procedure is iterative. This method is optimal but is not

complete, since the finite number of runs cannot guarantee that the robotic trajectories can fulfill the mission.

On the other hand, a parallel planning approach considers that both models, the robotic and the mission ones, are handled together, such that any of the robotic movements are synchronized with each constraint expressed by the mission model. Thus, this planning strategy is tackled throughout this thesis, considering a single composed model out of the Petri net representation of the robotic team and the Büchi automaton of the LTL mission. In both cases, the robotic trajectories are returned by solving mixed-integer linear programming (MILP) problems. The aim is to reduce the computational time to return a solution that ensures the given mission is fulfilled. This composed model is also extended towards time constraints, by integrating in the same model a Time Petri net model of the robotic team and a Timed automata associated with an MITL specification. In this scenario where time restrictions are included in the model, the motion solution is returned by model-checking approaches. The proposed strategy of joining two models into one has the benefit of taking into account the movement of heterogeneous robots, in contrast with [4] which is directed towards homogeneous robotic teams. Generally, the methods addressing solutions towards identical robots represent in some cases a limitation to the planning solutions that could be solved.

Another path planning strategy involves different classes of Petri net models, exploiting their advantages. Hence, hierarchical Petri net models, such as the Nets-within-Nets paradigm, have been proposed to handle the complexity of heterogeneous robotic teams by allowing tokens to represent other Petri nets, thus enabling a more flexible planning process [57, 58]. Since this paradigm has not been explored for motion planning solutions, improvements in the robotic representation design and computational efficiency remain a key focus of current research. To slightly close the gap, the Nets-within-Nets paradigm is investigated in this thesis for robotic planning, considering a proposed framework that is further evaluated on heterogeneous robotic team scenarios, respectively on homogeneous robotic teams. The latter case is compared also with other relevant DES methods from the literature.

Research in this field is continually advancing, with emerging solutions aimed the enabling robotic teams to efficiently and reliably execute complex tasks in both known and unknown environments. The planning strategies focus on two primary challenges: *(i) mitigating the state explosion problem and (ii) developing solutions for heterogeneous robotic teams.* In addressing these issues, the solutions proposed in this thesis are regarding Discrete Event Systems and high-level symbolic representations within the framework of temporal logic.

1.2 Problem description and contributions

The aim of the thesis can be summarized by the following problem statement:

Given a multi-agent system that should ensure a complex and rich mission under the temporal logic formalism, provide framework solutions based on Discrete Event Systems representations that allow for task assignment and collision-free trajectories.

The contributions of this PhD thesis tackle the identified challenges in the field of robotics, particularly focusing on advanced high-level planning methodologies, as mentioned in the problem statement. By addressing these complex issues, the proposed solutions offer innovative strategies to enhance autonomy, scalability, and versatility in robotic systems. Specifically, the contributions include:

- Task decomposition approach suitable for scenarios in which the team of robots is required to satisfy a complex mission which often tends to be computationally expensive [59]. Thus, decomposing a global mission into smaller independent tasks allows for easier handling of the problem and reduces the complexity by elevating properties such as modularity and adaptability. As far as it is presented in the literature, there is no automated technique that decomposes a high-level specification into independent tasks. Therefore, this gap is narrowed down by the proposed solution.

- Novel Petri net model based on the composition of two representations: one associated with the movement of the robotic team and one associated with the requirements that the robotic team should fulfill [35, 30, 60]. One downside of the sequential approach presented in the literature is represented by the iterative procedure since the robotic movement should follow a run guaranteeing the fulfillment of the mission. Therefore, the proposed framework allows to access the global state of the robots with respect to the mission. The proposed solutions enable the use of optimization problems, as well as model-checking methods.

- Introducing a framework under the Nets-within-Nets paradigm for motion planning a team of heterogeneous robots satisfying a global mission [61]. The novelty of this contribution lies in the defined formalism based on a hierarchical structure of Petri nets that encapsulates both local information about the states of the robots, as well as global information concerning the state of the robotic system with regards to the given high-level mission. The planning solution is returned through simulations, avoiding the state explosion problem present in the discrete event-based approaches. For this contribution, object-oriented properties are analyzed and incorporated into the proposed framework, leveraging the synchronization between the local and global data to fulfill the mission.

All the stated contributions aim to reduce the state explosion problem which was previously enunciated as being present in several planning strategies involving robotic teams.

In addition, the robotic movement solves the task assignments problem produced by the fact that in most cases, the robotic team ensures a global mission assigned to the entire team. Moreover, the latter contribution is directed towards exploring a particular class of Discrete Event Systems for the navigation problem of a heterogeneous multi-robot system, incorporating robots with different spatial capabilities when evolving in the workspace.

1.3 Thesis structure

The structure of this thesis is organized into several chapters, each addressing key components of the research. Illustrative examples accompany the theoretical formalism that addresses the motion planning problem as stated in the problem definition. Moreover, the proposed solutions detailed in this thesis are disseminated through scientific articles, as mentioned below. Thus, the remainder of the thesis details the following outline.

Chapter 2 introduces the fundamental concepts used throughout the thesis under the topic of Discrete event systems. These notions include the problem hypotheses considering the notations of robots, environments, and other key mathematical essential ideas. Afterward, a set of discrete event representations is defined, modeling the motion of the robotic team, as stated in the literature: *Transition systems* (TS), *Petri net* (PN), and *Time Petri net* (TPN). Additionally, various formalisms under temporal logic are defined for the high-level mission, such as *Linear Temporal Logic* (LTL) and *Metric Interval Temporal Logic* (MITL). In the end, a list of metrics and methods are described, which are considered in the comparison evaluation process between the proposed solutions and state-of-the-art.

Chapter 3 presents one planning approach including a task decomposition technique of the mission for a multi-robot system. The method decomposes a global Linear Temporal Logic mission, returning individual tasks that are assigned further to the robots, thus enhancing the autonomous behavior while reducing the synchronizations. The proposed method is evaluated through simulations in 3D environments, considering the motion of a team of drones, where the robotic model is based on a proposed 3D space partitioning algorithm, described formally in Chapter 2. This solution is published in:

- **Sofia Hustiu**, Ioana Hustiu, Marius Kloetzer, and Cristian Mahulea. LTL task decomposition for 3D high-level path planning. In *Journal of Control Engineering and Applied Informatics, 23(3), pp.76-87*, 2021.

Chapter 4 aims to provide a novel framework under the Petri net model, where the benefits of two representations are composed with the help of an intermediate layer. The representation encodes the behavior of the robotic team with respect to the workspace, respectively the global mission given to the team. Two composed models are defined. The framework is denoted *Composed Petri net model* and incorporates the motion of the robots ensuring spacial constraints given by the Linear Temporal Logic specification. The planning

strategy is based on solving optimization problems. In addition, this chapter proposes also an algorithm that ensures the robotic path execution through a parallel movement. The solution is evaluated in situations where the team of robots should pass through a narrowed free passage. The planning solution could lead also to a reallocation of tasks without interfering with the fulfillment of the mission. The work presented in this chapter includes results published in:

- **Sofia Hustiu**, Cristian Mahulea, Marius Kloetzer, and Jean-Jacques Lesage. On multi-robot path planning based on Petri net models and LTL specifications. In *IEEE Transactions on Automatic Control, vol. 69, no. 9, pp. 6373-6380,* 2024.
- **Sofia Hustiu**, Cristian Mahulea, and Marius Kloetzer. Parallel motion execution and path rerouting for a team of mobile robots. In IFAC-PapersOnLine, 55(28), 73–78. In *16th IFAC Workshop on Discrete Event Systems WODES,* 2022.

Chapter 5 proposes an extension to the *Composed Petri net model* from the previous chapter, by adding time constraints in both models: the Petri net model of the agent and the individual mission given as a Metric Interval Temporal Logic (MTIL) formula. Thus, the new framework is denoted *Composed Time Petri net* and it builds not only the spacial constraints but also temporal requirements with respect to the actions that the robots should ensure. This framework is flexible, being suitable to homogeneous robotic teams, respectively heterogeneous robotic teams, by incorporating the spatial capabilities of different robots ensuring individual missions, while the solution is provided by model-checking approaches. The proposed framework is published in two papers, as follows:

- **Sofia Hustiu**, Dimos V. Dimarogonas, Cristian Mahulea, and Marius Kloetzer. Multi-robot Motion Planning under MITL Specifications based on Time Petri Nets. In *2023 European Control Conference (ECC) (pp. 1-8). IEEE*, 2023.
- **Sofia Hustiu**, Alexandru-Florian Brasoveanu, and Andrei-Iulian Iancu. Integration of MITL for Cobots Workflow in a Manipulating Application. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8, 2024.

Chapter 6 explores the advantages of a particular model denoted *High-Level robot team Petri net*, embracing a hierarchical structure of Petri nets models, known under the name of *Nets within Nets paradigm*. Both the local and global states of each robot concerning the given global temporal logic mission are included in the same model. Moreover, this paradigm allows the use of object-oriented properties, making the model more versatile, since heterogeneous robots are modeled by the independent nets. The novelty of this work lies in the defined framework used for the motion planning problem, based on an established synchronization function that ensures the fulfillment of the mission when the robots have different spatial capabilities. The entire chapter is based on the following paper:

- **Sofia Hustiu**, Eva Robillard, Joaquín Ezpeleta, Cristian Mahulea, and Marius Kloetzer. Multi-robot Motion Planning based on Nets-within-Nets Modeling and Simulation. *Under review*. Available in [Online]: https://arxiv.org/abs/2304.08772, 2023.

Chapter 7 examines the results considering both numerical simulations and real applications. The evaluation of the proposed frameworks is compared with other Discrete Event Systems models present in the state-of-the-art, while the metrics include running time and length of the robotic trajectories ensuring the mission. The simulation captures scenarios with real-life use such as the automated whitening of greenhouses' roofs by a team of UAVs and assisting multi-robot systems in a hospital scenario, considering a team of both homogeneous and heterogeneous robots. The real experiments include the integration of individual MITL specifications for a manufacturing application. The results are present in three published papers, out of which two of them appear in the contributions of Chapter 5 (second paper) and 6.1:

- **Sofia Hustiu**, Marius Kloetzer, Eva Robillard, Alejandro López-Martínez, and Cristian Mahulea. Whitening of greenhouse's roof using drones and Petri net models. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, 2022.

Chapter 2

Discrete event systems for multi-agent path planning solutions

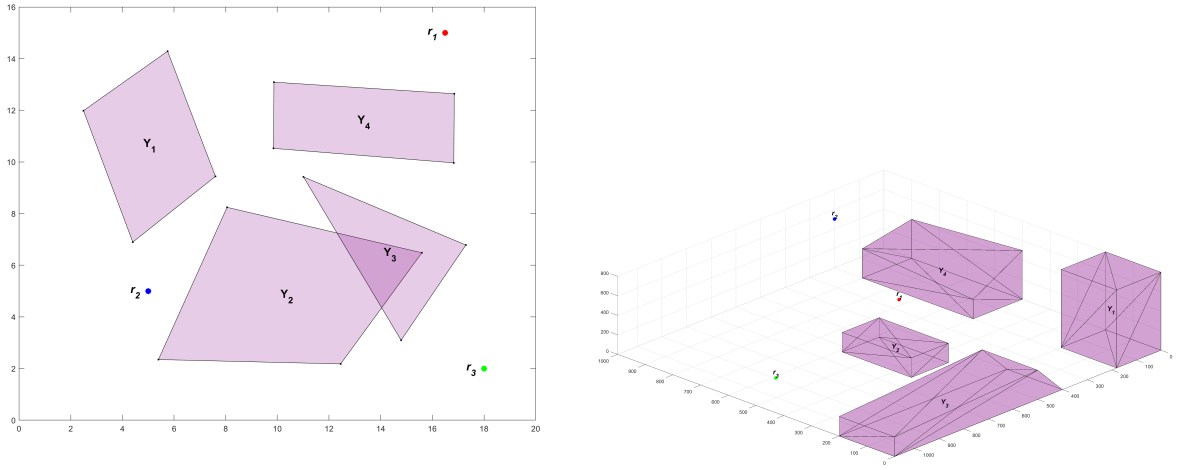
This chapter introduces the fundamental concepts underlying the representations used in the proposed path planning solutions for a multi-agent system tasked with completing a specific mission. Initially, a set of basic notions is defined to familiarize the reader with the context of a multi-robot system operating within a given workspace. Following this, two models are formally presented within the framework of discrete event systems, characterizing the high-level motion of a robotic team in relation to the spatial constraints of the environment. Finally, several approaches to mission description are detailed, incorporating theoretical notations that impose spatial and temporal restrictions on the team's movement. Additionally, these formalisms facilitate the sequencing and synchronization of actions. Each concept is illustrated with examples, demonstrating the relationship between theory and its practical application.

2.1 Problem hypotheses

This section introduces several notions to facilitate understanding of the subsequent theoretical definitions. These foundational concepts are crucial for the reader to comprehend the modeling of a robotic team and its assigned mission.

2.1.1 Workspace of the robotic system

Let us consider a set of mobile robots, often referred to as *agents* to broaden the applicability of the proposed method across different domains, denoted by $\mathscr{R} = \{r_1, r_2, \ldots, r_{|\mathscr{R}|}\}$. This set symbolizes robots that are assumed to be punctiform and omnidirectional, evolving in either 2D, as well as 3D spaces (for example, modeling UAVs, known also as drones). These robots operate within a known environment E, which contains several regions of interest (ROIs)



(a) 2D environment (b) 3D environment

Fig. 2.1. Example of a workspace with four regions of interest for a team of three robots

defined by the set $\mathscr{Y} = \{y_1, y_2, \ldots y_{|\mathscr{Y}|}\}$. These regions are further of interest for the robotic team, as they can represent both regions that should be reached, e.g., a room in a building, or regions that should be avoided, e.g., a tree in a forest. Considering various applications for robotic systems, the regions are assumed to be either disjoint or partially overlapped. Figure 2.1 illustrates two examples of working spaces containing 4 regions of interest (purple color), and a team of 3 robots, illustrated with colored bullets: r_1-red, r_2-blue, and r_3-green. Figure 2.1(a) exemplifies a 2D environment, where regions y_2 and y_3 are overlapped, while Figure 2.1(b) shows a 3D environment with all 4 regions, which do not overlap to provide a clearer picture. As observed, the regions can have different shapes. For the 3D environment, all regions have a flat base surface.

The environment E can be represented through a discrete space representation associated with the continuous space for easier manipulation of the working space in which the robots evolve. The partitioning methods presented in the literature associate a discrete set of elements with the relevant features inside E, such as the free space or the set of regions of interest, i.e., $\mathscr{Y}$. Figure 2.2 presents a classification of the mapping techniques, also encountered in 3D spaces based on the extension of existing 2D methods.

The information sensed by a robot in the environment is represented in a map, to facilitate an easier deployment in free space. Considering the classification of the mapping techniques illustrated in Figure 2.2, the first three algorithms (light blue color) output a discrete space representation which can be further handled via a graph or an automata model. Therefore, a path planning method applied to a robot moving in a continuous environment can be returned by graph search-based algorithms such as Dijkstra, A*, or others [62] addressed to the graph representation. These techniques are suitable for offline planning, providing

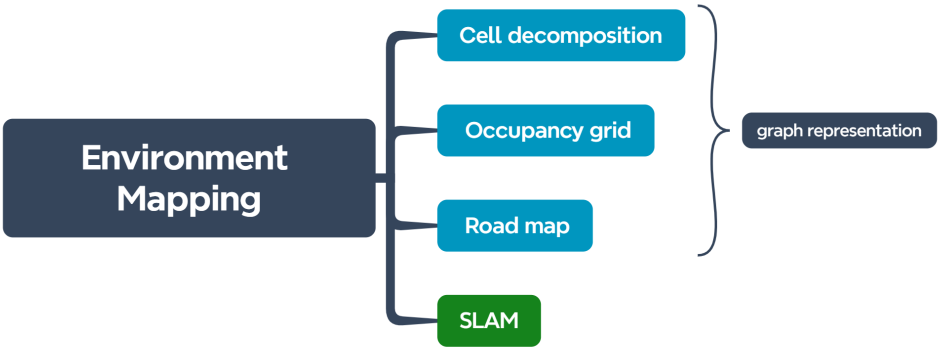


Fig. 2.2. Classification of mapping techniques (blue - mapping, green - mapping and localizing)

a clear picture of the free space, especially for static environments. On the other hand, the SLAM (Simultaneous Localization and Mapping) [63] algorithm generates an online map as a result of the surrounding sensing capabilities of a robot, e.g., through the on-board cameras or radars.

The challenges of a mapping technique remain the *accuracy of the map, real-time localization of the robot, and the scalability of the mapping technique*, especially in new scenarios such as dynamic environments. The mentioned techniques map the free space using discrete approaches (cell decomposition, road map), stochastic approaches (occupancy grid), or real-time approaches (SLAM).

2.1.2 Cell decomposition techniques for Discrete Event Systems

In this thesis, the mapping method that associates a discrete space representation of the continuous working space is based on the cell decomposition approach. By employing this method in either 2D, either 3D environments, the visualization of the partitioned workspace is enhanced. Moreover, this partitioning method allows for an easier manipulation of the environment concerning also the number of robots in the team and their dynamics. The main idea is to divide the working space into a set of disjoint polytopes (polygons, respectively cuboids for 2D, respectively 3D environments). Depending on the mapping techniques, this space discretization can be precise (capturing the entire free space) or approximately (capturing almost all the entire free space). Let us denote with $\mathscr{C} = \{c_1, c_2, \ldots c_{|\mathscr{C}|}\}$ the set of cells represented by the disjoint polytopes. A brief introduction to these decomposition techniques is presented in the following, considering the 2D environments. A detailed description of these algorithms can be examined in [1, 15]. The 3D partitioning method is denoted *3D rectangular cuboid decomposition* and it is further detailed, since the algorithm represents a personal contribution for this thesis.

(a) *Triangular decomposition* - This method uses triangular cells formed by a specific triangulation technique, not Delaunay triangulation. Particularly, no edge of any cell does not crosses the regions of interest, as it would happen through a classic Delaunay triangulation. Thus, the environment is composed of non-overlapping triangles avoiding intersecting obstacles by using predefined line segments from obstacle facets and the environment's vertices [64].

(b) *Trapezoidal decomposition* - Trapezoidal cells are formed by extending vertical lines from each vertex to the facets in both directions (up and down). When these lines intersect facets or the environment boundary, new trapezoids are defined [65].

(c) *Polynomial decomposition* - This decomposition forms polytope-shaped cells with varying vertices by extending each obstacle facet to the environment's border. The convex polygon cells collectively cover the entire free space.

(d) *Rectangular decomposition* - This decomposition, inspired by quad-trees, uses nodes that are either leaves or have four children. The main idea is to recursively divide each cell containing an obstacle by splitting each ax in half until a given precision is reached. The precision indicates the smallest cell that can be derived, considering that all cells maintain the length and width ratio with the workspace [66].

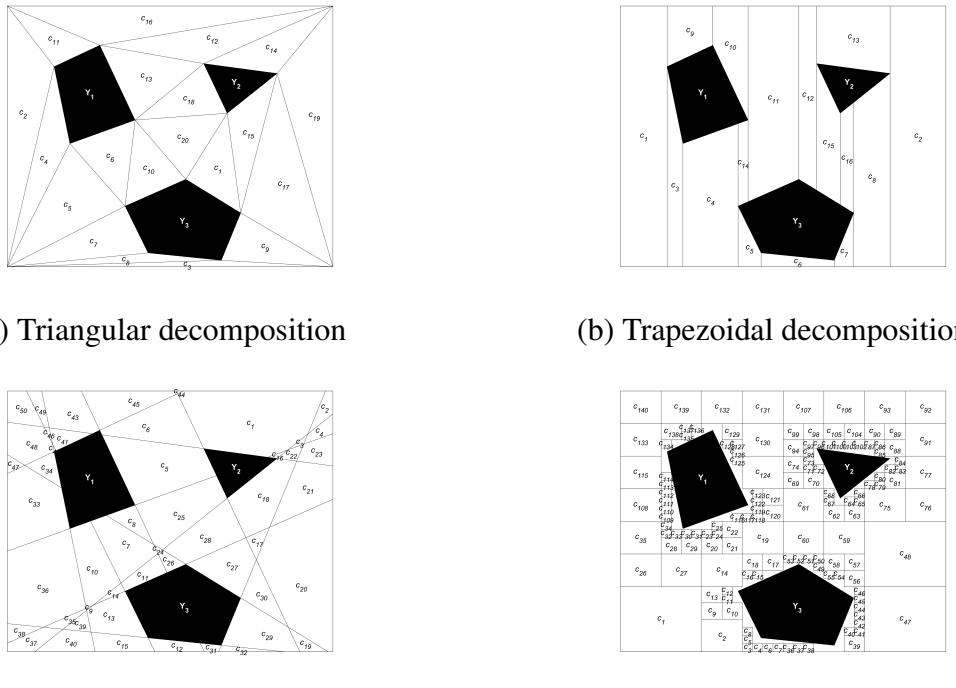
Figure 2.3 portrays the previously described partitioning approaches. Considering the examples and the short description of these methods from [1], the regions of interest visualized with the black color represent obstacles for the robotic team. Therefore, the cell decomposition method maps fully or partially the free space. The cells are numbered in the order they appear, with the number displayed at the centroid of each cell, e.g., c_1. Throughout the thesis, the partitioning techniques are applied to the entire environment, and the regions of interest are divided into cells. The following algorithm for the 3D cell decomposition is accompanied by explanations and illustrative examples considering this scenario.

3D rectangular cuboid decomposition

Some decomposition methods are extended to 3D environments. The (a) triangular decomposition presents several alternatives for 3D spaces as presented in [67]. The (d) rectangular decomposition can be extrapolated towards three axes, known under the name of *rectangular cuboid decomposition* [68, 69], which can be divided into two approaches:

(i) *Grid cell decomposition* - the resulting partitioned workspace E is based on a division of rectangular cuboids which are equal. Therefore, a precision ε is required to be given, computing the number of equal cuboids (cells). For example, for $\varepsilon = 8$, the whole environment is divided into 8^3 equal rectangular cuboids.

(ii) *Oct-tree based decomposition* - the imposed precision ε is used to recursively divide the environment E space into rectangular cuboid of different sizes. Thus, the cells



(a) Triangular decomposition

(b) Trapezoidal decomposition

(c) Polytopal decomposition

(d) Rectangular decomposition

Fig. 2.3. Various methods of cell decompositions [1]

maintain the same width, length, and height as the given environment. Generally, this method is used when an increased resolution (smaller cells) is needed since the partitioned workspace returns fewer cells than in the previous method.

The rectangular decomposition of Figure 2.3 is particularized for the scenario in which the partitioning method maps almost the entire free space of the environment since the edges of the cells do not overlap with the edges of the black regions. As mentioned previously, both the 2D and the 3D partitioning methods are generally applied for the entire working space, dividing also the regions of interest into cells. This principle is also applied to the 3D rectangular decomposition. An example of enabling this decomposition technique is in [70], a paper that addresses a planning strategy for a drone in a dynamic environment.

The Algorithm 1 presents one of the contributions of this thesis, detailing the recursive procedure of partitioning the 3D working space into rectangular cuboid decomposition. The 3D decomposition from the approach in [69] aims to integrate the sensor's reading into the algorithm based on probabilistic occupancy estimation in order to provide flexibility towards the mapping technique. Compared with this work, the current decomposition method provides a generalized step-by-step algorithm that divides the environment recursively into cells. Moreover, these cells are labeled into: *occupied* if all points within it lie entirely within a single or a combination of multiple regions of interest within the set $\mathscr{Y}$, that the multi-agent

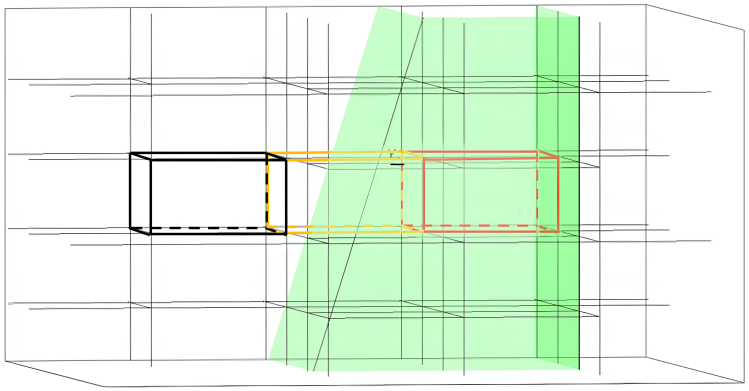


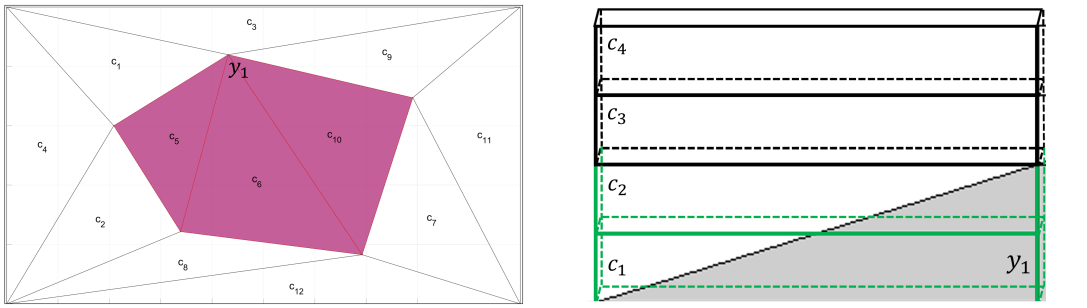
Fig. 2.4. Enlarged part of the 3D partition obtained by Algorithm 1, showing a free cell (black), a mixed cell yellow, and an occupied one (red).

system should reach or avoid, *free* if none of its points are encompassed by any obstacle; *mixed* if it contains points that are within a region as well as points that are not within the region.

Before introducing a labeling function necessary for the formal notation used in the thesis, let us first introduce the notion of atomic proposition. An *atomic proposition* $b_i \in \mathscr{B}$ represents a variable with Boolean values: True ($\top$) or False ($\bot$), with $\mathscr{B}$ being the set of atomic propositions. In the following, let us refer to an atomic proposition with *observation*, that can be *active* (if is evaluated as True), or *inactive* (if is evaluated as False). Each atomic proposition is assigned to each region of interest, the mapping relation being one-to-one. Thus, a labeling function can be defined: $h : \mathscr{C} \to 2^{\mathscr{B}}$, with $\mathscr{C}$ being the set of cells resulting from the decomposition method, $2^{\mathscr{B}}$ being the power set of the set $\mathscr{B}$, associated further with the set $\mathscr{Y}$. The notation $\emptyset \in 2^{\mathscr{B}}$ represents the label for the elements included in the free space.

Example 2.1.1 *Figure 2.4 represents an enlarged part of a 3D rectangular cuboid decomposition, showing a free cuboid with a black color, a mixed cell with a yellow color, and an occupied one with red color (this being completely included in the region y_1). For the entire environment, the partitioning method considered a precision of $\varepsilon = 16$ (the maximum number of divisions for each ax).* ■

Remark 2.1 Since two cells might indicate towards the same region of interest y_i, one of the cells being *occupied*, while the second one being *mixed*, the Algorithm 1 necessitates additional information to differentiate between these two cells. Therefore, a function $\alpha : \mathscr{C} \to \{-1, 0, 1\}$ is added for all the cells, with the values assigned in order of the cells *mixed, free, occupied*. The standalone value assigned through function h is not sufficient to differentiate these two mentioned labels.



(a) 2D polytopal decomposition (b) 3D rectangular cuboid decomposition

Fig. 2.5. Examples of cells labeled as region of interest

First, the entire environment is considered as the current cell *RC* (line 1). The next line computes several volumes: of the current cuboid *RC*, of the environment, and V_i considering the intersection between *RC* and every region of interest $y_i \in \mathscr{Y}$. For the intersection volume, the half-space (H-) representation is used, [71] (lines 2-3). The test from line 5 is fulfilled only for *free* cells, which are added to set $\mathscr{C}$. For example, if no region of interest is in the environment, then the environment is composed of only one free cell. Otherwise, if the current cell *RC* partially intersects at least one region from $\mathscr{Y}$ and the cell is bigger than the imposed precision ε (test on line 10), then *RC* should be divided into smaller cells. Specifically, *RC* is divided into 8 smaller cuboids based on the procedure *recursive_partitioning* follows (lines 10-18). Once the precision is reached, the cells are labeled as *mixed* if the cell is partially included in at least one region of interest (lines 25-27), *occupied* if the cell is fully included in at least one region of interest (lines 29-30), and *free* otherwise (lines 32-33). In this thesis, the algorithms applied to partition the environment are the 2D polynomial and rectangular decomposition methods, respectively the 3D rectangular cuboid decomposition. The following chapters highlight the applicability of the mixed and occupied cell type to be a region of interest for the team of robot, that shall be reached.

Example 2.1.2 *To enhance the understanding of the fundamental concepts discussed in this chapter, let us examine a simple example illustrated in Figure 2.5. The left side of the figure demonstrates a polytopal decomposition of the environment, resulting in a total of 12 cells. Conversely, the right side of the figure depicts a partial 3D rectangular cuboid decomposition consisting of 4 cells.*

In the polytopal partitioning, the edges of the region y_1 *(highlighted in magenta) are utilized as the edges for the cells. Consequently, it can be observed that three cells are designated as regions of interest:* $h(c_5) = h(c_6) = h(c_{10}) = b_1$ *and* $mixed(c_5) = mixed(c_6) = mixed(c_{10}) = 1$*, with* b_1 *being the atomic proposition for* y_1*, whereas the remaining cells are designated as free, e.g.,* $h(c_1) = \emptyset$*. As previously mentioned, in the 3D rectangular cuboid method, cells labeled as regions of interest can either be mixed or occupied cells. In this example, the mixed cuboids are labeled with* b_1 *(region* y_1 *(indicated in gray), specifically*

Algorithm 1: 3D rectangular cuboid decomposition

Input : Environment E, regions of interest $\mathscr{Y}$, set of atomic propositions $\mathscr{B}$, precision ε

Output : Set of cells $\mathscr{C}$, labeling function h and α function

Denote $RC = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$ modeling E as the current rectangular cuboid

$V_{RC} = Volume(RC), V_E = Volume(E), V_i = Volume(RC \cap y_i), \forall i = 1, \ldots, |\mathscr{Y}|$

$V_{int} = \sum_{i=1}^{|\mathscr{Y}|} V_i$

$(\mathscr{C}, h, \alpha) = recursive\_partitioning(RC, \mathscr{C}, h, \alpha, \varepsilon, \mathscr{Y})$

if $V_{int} = 0$ **then**

  /* the current rectangular cuboid RC is *free* */

  $\mathscr{C} = \mathscr{C} \cup RC$, $h(RC) = \emptyset$, $\alpha(RC) = 0$

else

  if $V_{RC} > V_E/\varepsilon^3$ **then**

    /* RC is bigger than the imposed ε and overlaps at least one region $y_i \in \mathscr{Y}$ */

    $(\mathscr{C}, h, \alpha) = recursive\_partitioning([x_{min}, \frac{x_{min}+x_{max}}{2}] \times [y_{min}, \frac{y_{min}+y_{max}}{2}] \times [z_{min}, \frac{z_{min}+z_{max}}{2}], \mathscr{C}, h, \alpha, \varepsilon, \mathscr{Y})$

    $(\mathscr{C}, h, \alpha) = recursive\_partitioning([x_{min}, \frac{x_{min}+x_{max}}{2}] \times [\frac{y_{min}+y_{max}}{2}, y_{max}] \times [z_{min}, \frac{z_{min}+z_{max}}{2}], \mathscr{C}, h, \alpha, \varepsilon, \mathscr{Y})$

    $(\mathscr{C}, h, \alpha) = recursive\_partitioning([x_{min}, \frac{x_{min}+x_{max}}{2}] \times [y_{min}, \frac{y_{min}+y_{max}}{2}] \times [\frac{z_{min}+z_{max}}{2}, z_{max}], \mathscr{C}, h, \alpha, \varepsilon, \mathscr{Y})$

    $(\mathscr{C}, h, \alpha) = recursive\_partitioning([x_{min}, \frac{x_{min}+x_{max}}{2}] \times [\frac{y_{min}+y_{max}}{2}, y_{max}] \times [\frac{z_{min}+z_{max}}{2}, z_{max}], \mathscr{C}, h, \alpha, \varepsilon, \mathscr{Y})$

    $(\mathscr{C}, h, \alpha) = recursive\_partitioning([\frac{x_{min}+x_{max}}{2}, x_{max}] \times [y_{min}, \frac{y_{min}+y_{max}}{2}] \times [z_{min}, \frac{z_{min}+z_{max}}{2}], \mathscr{C}, h, \alpha, \varepsilon, \mathscr{Y})$

    $(\mathscr{C}, h, \alpha) = recursive\_partitioning([\frac{x_{min}+x_{max}}{2}, x_{max}] \times [\frac{y_{min}+y_{max}}{2}, y_{max}] \times [z_{min}, \frac{z_{min}+z_{max}}{2}], \mathscr{C}, h, \alpha, \varepsilon, \mathscr{Y})$

    $(\mathscr{C}, h, \alpha) = recursive\_partitioning([\frac{x_{min}+x_{max}}{2}, x_{max}] \times [y_{min}, \frac{y_{min}+y_{max}}{2}] \times [\frac{z_{min}+z_{max}}{2}, z_{max}], \mathscr{C}, h, \alpha, \varepsilon, \mathscr{Y})$

    $(\mathscr{C}, h, \alpha) = recursive\_partitioning([\frac{x_{min}+x_{max}}{2}, x_{max}] \times [\frac{y_{min}+y_{max}}{2}, y_{max}] \times [\frac{z_{min}+z_{max}}{2}, z_{max}], \mathscr{C}, h, \alpha, \varepsilon, \mathscr{Y})$

    else

      /* the current cuboid RC reached the imposed precision ε */

      $\mathscr{C} = \mathscr{C} \cup RC$, $V_i = Volume(RC \cap y_i), \forall y_i \in \mathscr{Y}$

      $h(RC) = \{b_i \in \mathscr{B} \,|\, V_i > 0\}$ OR $h(RC) = \{\emptyset \,|\, V_i = 0\}$

      if $y_i \in h(RC)$ **then**

        /* region y_i overlaps with RC */

        if $V_i < V_{RC}$ **then**

          /* RC is labeled as *mixed* */

          $\alpha(RC) = -1$

        else

          /* RC is labeled as *occupied* */

          $\alpha(RC) = 1$

      else

        /* RC is labeled as *free* */

        $\alpha(RC) = 0$

$h(c_1) = h(c_2) = b_1$, *and* $mixed(c_1) = mixed(c_2) = -1$, *while* $mixed(c_3) = mixed(c_4) = 0$.
■

In order to provide a comprehensive overview of the environment and the notations used throughout this thesis, it is essential to also introduce the following definitions ([72]), required for the representations of the robotic team and the mission.

- An *infinite word* over a set $\mathscr{B}$ is an infinite sequence $r = b_1 b_2 \ldots$, where $b_i \in \mathscr{B}$ is the i-th element of the sequence.
- A *time sequence* is an infinite sequence of time, denoted by $\tau = \tau_0 \tau_1 \ldots$, where $\tau_i \in \mathbb{R}_+$ and has the following properties: *monotonicity:* $\tau_i < \tau_{i+1}$ for all $i \geq 0$; *progress:* for all $t \in \mathbb{R}_+$, there exists $i \geq 1$ such that $\tau_i > t$.
- A *timed word* over a set $\mathscr{B}$ is defined as an infinite sequence $w^t = (b_1, \tau_0)(b_2, \tau_1) \ldots$, where $b_1 b_2 \ldots$ is an infinite word and $\tau_0 \tau_1 \ldots$ is a time sequence.

Remark 2.2 The set $\mathscr{B}$ of atomic propositions is associated with the regions of interest $\mathscr{Y}$ that the robotic team should reach and/or avoid as stated in the global mission. Throughout this thesis, the set of atomic propositions will be also represented as the set of actions to be executed in those regions, such as picking up an object or cleaning a room, i.e., an atomic proposition b_i corresponds to the action i (Chapter 5). Let us consider a subset $A \in \mathscr{B}$. For this subset, let us define $A_\wedge = \bigwedge\{b_i \in \mathscr{B}$ as being the characteristic conjunction formula of A. For example, for $A = \{b_1, b_2, b_3\}$, $A_\wedge = b_1 \wedge b_2 \wedge b_3$, using the set of atomic propositions $\mathscr{B}$.

As a short overview of the notions defined previously, let us resume the key ideas. A 2D, respectively a 3D environment including a continuous space can be associated with a discrete space representation based on a cell decomposition technique. Thus, the partitioned environment is represented by a set of discrete elements $\mathscr{C}$, denoted cells. These cells capture both the free space of the environment as well as the regions of interest $\mathscr{Y}$ that the team of robots should reach and/or avoid as part of their plan. Multiple cells can map a single region of interest, e.g., cells c_1 and c_2 are part of the same region of interest y_i. The labeling of the cells is based upon the set of atomic propositions $\mathscr{B}$ through function h, having a one-to-one relation to the regions of interest from set $\mathscr{Y}$.

2.2 Discrete event agent representations

This subsection aims to provide the elementary notions considered for modeling the high-level motion of the robotic team under the class of discrete event systems (DES). These systems offer insights into where a state change occurs in discrete time triggered by events. These systems are characterized by a sequence of instantaneous events rather than continuous flows of time [73]. Several examples include robotic control [1] and manufacturing [74] systems.

In the context of robot path planning, DES can be used to model the sequences of actions and decisions robots must make to move through an environment. Each event in the DES corresponds to a specific action or decision point for the robot, such as moving to a new location, picking up an object, or avoiding an obstacle. By using DES, the path planning process can be structured as a series of continuous and/or discrete steps, making it easier to manage complex tasks. This approach helps in designing algorithms for robot movement and task execution. In essence, the continuous space in which the robots evolve is first partitioned into a set of elements capturing the free space and the regions that should be reached or avoided. This set can be abstracted as one discrete event system model for the robotic team afterward. Throughout the thesis, the representation of the robotic model is based on Petri net models, with or without time (as it will be formally defined in the following). The model's topology is built upon the discrete space representation in which the robotic team evolves, while tokens represent the robots. One advantage of the Petri net representations consider the update in its structure to encapsulate constraints, e.g., collision avoidance, one solution being to impose a maximum capacity (how many robots) that can be located in a particular space from the workspace [75]. The following illustrative examples accompany the theoretical fundamental notion defined, to enhance the benefits of the Petri net representations in the robotic path planning strategies.

Example 2.2.1 *Figure 2.6 illustrates an example of a partitioned workspace in 9 cells. The working space consists of three regions of interest: y_1 representing the trees, y_2 representing the hydrant, and y_3 defining the fire. An atomic proposition is defined for each region: b_1 for y_1, b_2 for y_2, and b_3 for y_3. The observation function h assigns to each cell an atomic proposition, for easier manipulation of the partitioned space: $h(c_2) = h(c_3) = b_1, h(c_7) = b_2, h(c_3) = b_3$, while the rest of the places model the free space $h(c_1) = h(c_4) = h(c_6) = h(c_8) = h(c_9) = \emptyset$.* ■

Considering the partitioned environment into cells, the following definition presents the components of an agent's representation, including the analogy between each element in the definition and the physical workspace where the agent evolves. The introduced models include several basic concepts detailed in [1, 76].

2.2.1 Petri net model

Other DES representations are defined below to facilitate easier model manipulation for a multi-agent system.

Definition 2.2.2 *[1] A Robot Motion Petri Net system (RMPN) is characterized by the tuple $\mathscr{Q} = \langle \mathscr{N}, m_0, \mathscr{B}, h \rangle$, where:*

- $\mathscr{N} = \langle P, T, Post, Pre \rangle$ *is a Petri net composed of:*

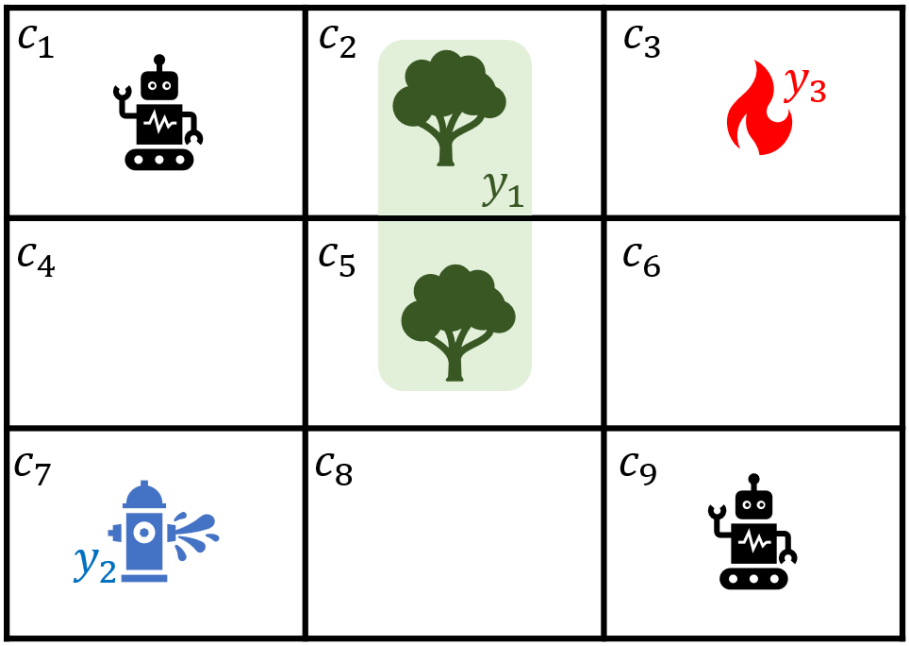


Fig. 2.6. Workspace of a partitioned environment with a team of two robots

  - *A set of places P (each place representing a cell in the set $\mathscr{C}$);*
  - *A set of transitions T, each signifying a robot's movement between neighboring cells;*
  - *$Post \in \{0,1\}^{|P|\times|T|}$, the post-incidence matrix, detailing the connections from transitions to places. Specifically, $Post[p,t] = 1$ if transition $t \in T$ is connected to place $p \in P$, and $Post[p,t] = 0$ otherwise;*
  - *$Pre \in \{0,1\}^{|P|\times|T|}$, the pre-incidence matrix, outlining the connections from places to transitions. Specifically, $Pre[p,t] = 1$ if place $p \in P$ is connected to transition $t \in T$, and $Pre[p,t] = 0$ otherwise;*

- *m_0 is the initial marking vector, where $m_0[p]$ denotes the number of robots initially located in cell $c \in \mathscr{C}$;*

- *$\mathscr{B} \cup \{\emptyset\}$ represents the set of output symbols represented by the atomic propositions and associated with the set of regions if interest $\mathscr{Y}$;*

- *$h : P \to 2^{\mathscr{B}}$ is the observation function. If place p_i contains at least one token (indicating the presence of at least one robot in cell c_i), then the region(s) of interest associated with the atomic propositions of set $\mathscr{B}$ is (are) considered as visited and marked in $h(p_i)$.*

Note that the observation function h has been previously defined on the set of elements $\mathscr{C}$ ($h : \mathscr{C} \to 2^{\mathscr{B}}$). Since the discrete space representation is further represented by the RMPN

model, where each place $p \in P$ denotes a cell $c \in \mathscr{C}$, the function h will be defined on the set P throughout this thesis.

The total number of tokens of the RMPN model is equal to $|\mathscr{R}|$. Notably, this ensures that the model's structure (comprising the number of places, transitions, and arcs) remains invariant even when robots are added to or removed from the team. Only the marking (state) of the RMPN is altered. Notice that a difference between the robots cannot be determined, as a token is associated with one robot. Thus, this representation is suitable for homogeneous robotic teams.

By definition, the RMPN systems addressed in this thesis belong to the class of state machines, where each transition has one input and one output place. For a given transition $t_j \in T$, ${}^\bullet t$ represents its input place, and $t^\bullet$ denotes its output place. Formally, the notations are defined as ${}^\bullet t_j = \{p_i \in P | Pre[p_i, t_j] = 1\}$ and $t_j{}^\bullet = \{p_i \in P | Post[p_i, t_j] = 1\}$. Transition $t_j \in T$ is enabled at marking m if its input place contains at least one token i.e., $m[p_i] \geq 1$, where $p_i = {}^\bullet t_j$.

In the RMPN, all arcs have a weight equal to one. In a general PN (with arc weights greater than one), a transition is enabled if the number of tokens in its input place is greater than or equal to the weight of the arc that connects the place with the transition.

When an enabled transition t_j fires, the RMPN reach a new marking $\tilde{m} = m + C[\cdot, t_j]$, where m is the initial marking, and $C = Post - Pre$ is the token flow matrix, with $C[\cdot, t_j]$ denotes the column corresponding to t_j. In the RMPN context, firing a transition t_j corresponds to a robot moving from cell ${}^\bullet t_j$ to cell $t_j{}^\bullet$. For the moving robot, transition t_j involves executing a control law that directs the robot from cell ${}^\bullet t_j$ to $t_j{}^\bullet$, and there are established methodologies for developing such continuous control laws in specific scenarios [77, 78].

Example 2.2.3 *The RMPN model associated with the environment described in Example 2.2.1 is present in Figure 2.7. The model contains 9 places and 24 transitions, representing the discrete space representation illustrated in Figure 2.6. The adjacency relation between two cells resulted from the cell decomposition methods, c_i, c_j with $i, j = \overline{1,9}, i \neq j$ is represented by two transitions in the RMPN model: one indicating the movement of the robot from c_i to c_j, while the second one indicating the movement from c_j to c_i, i.e., the pair t_1 and t_2 for places p_1 and p_2. The Pre and Post matrices for this RMPN model are expressed as follows:*

$$Pre = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{bmatrix}$$

$$Post = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}$$

Visually, places p_2, p_5 have the green edge color since these two places model the region y_1: $h(p_2) = h(p_5) = b_1$. Respectively, place p_7 portrays the region y_2 (blue color): $h(p_7) = b_2$, while place p_3 is associated with region y_3 (red color): $h(p_3) = b_3$. The color coding is similar to the one portrayed in Figure 2.6

The initial position of the identical robots is expressed by the marking vector $m_0[p_1] = m_0[p_9] = 1$, since the robots are initially in cells c_1 and c_9, modeled here by tokens included in the places p_1, p_9. The marking for the rest of the places is equal to 0, i.e., $m_0[p_4] = 0$. ■

The goal is to identify sequences of transitions that need to be fired to ensure the team meets a desired configuration in the workspace, translated into a desired marking. If the desired marking $\tilde{m}$ can be reached from a marking m through a finite sequence of transitions σ, the firing count vector is denoted by $\sigma \in \mathbb{N}_{\geq 0}^{|T|}$, where the j^{th} element indicates the cumulative number of firings of t_j. In this context, the state (or fundamental) equation is expressed as:

$$\tilde{m} = m + C \cdot \sigma \tag{2.1}$$

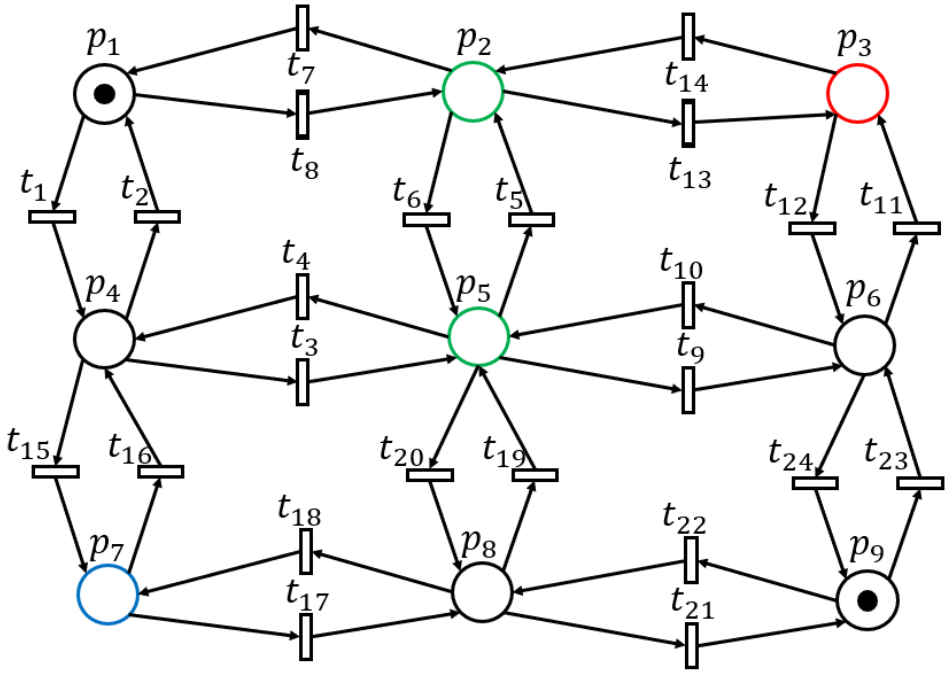


Fig. 2.7. The RMPN model associated with the robotic team for the given environment from Example 2.2.1

A firing vector σ with a minimal number of transitions can be determined to drive the live[1] RMPN to the desired marking $\tilde{m}$ by solving an optimization problem with the cost function $1^T \cdot \sigma$. The details specifying the methodology of converting a firing vector into a sequence of robot movements are described in [1].

As stated in [1], the equation (2.1) is only a necessary condition for the reachability of a marking. The marking solutions that are not reachable are referred to as *spurious markings*. In general, determining whether a marking m is reachable or not, is challenging due to these spurious markings. Another issue that can appear by searching the minimum sequence of transition of reaching a desired marking $\tilde{m}$, is to avoid the looping procedure.

An intuitive scenario for this situation can be described by recalling the Example 2.2.3. The marking $\tilde{m}[p_1] = \tilde{m}[p_8] = 1$, with $\tilde{m}[p_i] = 0$ for $i \neq 1, i \neq 8$, can be reached: (i) by firing one time transition t_{22}, i.e., $\sigma[t_{22}] = 1$, with $\sigma[t_i] = 0, i \neq 22$; (ii) by firing twice the transition t_{22} and one time transition t_{21}, i.e., $\sigma[t_{22}] = 2$, $\sigma[t_{21}] = 1$ with $\sigma[t_i] = 0, i \neq 22, i \neq 21$. The later situation generates a loop sequence for the robot, which can also be repeated multiple times.

[1]A Petri net is considered live if, regardless of the currently reachable marking, all transitions can eventually fire.

2.2.2 Time Petri net model

The succeeding representation allows the robotic team to incorporate time constraints directly into the motion planning, by specifying timing requirements to move from one cell to another. This characteristic is essential for coordinating the movements of robots. The formal definition of *Time Petri net* model is based upon the theoretical notions from [79–81] but extends the mathematical notation used for RMPN.

Definition 2.2.4 *A Time Petri net (TPN) model is defined as a tuple* $\mathscr{TPN} = \langle \mathscr{Q}, I \rangle$*, where:*

- $\mathscr{Q} = \langle \mathscr{N}, m_0, \mathscr{B}, h \rangle$ *denotes the previously defined RMPN model (Definition 2.2.2);*
- $I : T \rightarrow [\mathbb{Q}_+ \rightarrow \mathbb{Q}_+ \cup \{\infty\}]$ *maps a static interval to each transition. The time interval for a transition is represented by a tuple* $I(t) = [\alpha, \beta]$ *for all* $t \in T$*, where* $0 \leq \alpha < \infty$ *denotes the earliest firing time,* $0 \leq \beta \leq \infty$ *denotes the latest firing time, and* $\alpha \leq \beta$ *if* $\beta \neq \infty$*, or* $\alpha < \beta$ *if* $\beta = \infty$*.*

Simply put, the Time Petri net representation is built upon the RMPN model, by adding a time duration to the set of transitions, related to the motion of the robots throughout the environments. As stated in [81], this representation can be expressed as an RMPN, where a clock is assigned for each transition. Since the Time Petri net model includes the time dimension, a particularity of this model in contrast with the RMPN is that a state is characterized by a pair given by the marking m as defined previously, and another function that expresses the clock for each transition [81]. Thus, the state of the Time Petri net is triggered by elapsing a time that, if is smaller than a time upper bound of the enabled transitions, could fire one of them. This action updates both the marking of the net, as well as time that passed. The detailed procedure of enabling transitions, firing them, and updating the state of the Time Petri net is formally defined in [81], and illustrative examples are given in [82].

To understand better how the firing of transitions is conducted, let us consider a small example, based on Figure 2.8. Considering the marking in $m[p_9] = 1$, then the transition t_{22} is enabled. This transition fires when the time elapsed is between the boundaries δ^{22}_{min} and δ^{22}_{max}. Hypothetically, let us assume that another transition t_{23} is connected to the input place p_9 and an output place p_6. Therefore, two scenarios could be described: (i) $\delta^{23}_{max} > \delta^{22}_{max}$ and (ii) $\delta^{23}_{max} < \delta^{22}_{max}$. Let us first consider that a time Δ_1 elapsed, with $\Delta_1 < \delta^{22}_{max}, \Delta_1 < \delta^{23}_{max}$. Thus, both transitions t_{22}, t_{23} are enabled. Moreover, the state of the Time Petri net updates with time Δ_1, for the same marking $m[p_9] = 1$. Specifically, the current lower bound for t_{22} could be now considered equal with $\delta^{22}_{min} + \Delta_1$, respectively $\delta^{23}_{min} - \Delta_1$ for t_{23}. In this case, any of the transitions could be fired, updating also the marking in the new state of the model. Note that Δ represents a time elapsed with respect to the global time, while the time interval associated with each transition considers a local clock.

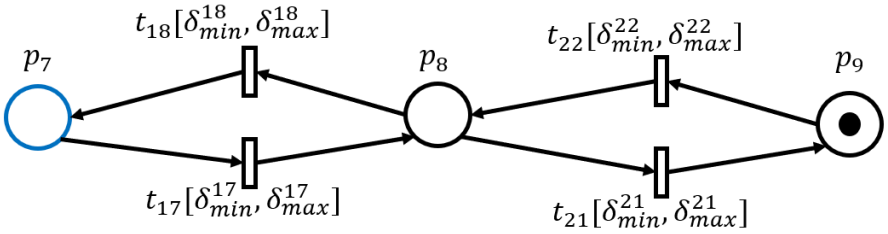


Fig. 2.8. Representation of a discrete workspace into a time Petri net model

If another time Δ_2 passes, with $\Delta_2 + \Delta_1 \geq \delta^{22}_{max}$ and $\Delta_2 + \Delta_1 < \delta^{23}_{max}$, then the first scenario (i) occurs, and transition t_{22} is forced to fire. This action leads to a new state, with a new marking $m[p_8] = 1$. Moreover, the local clock for t_{23} restarts, considering again the lower bound δ^{23}_{min}. Contrary, if $\Delta_2 + \Delta_1 \geq \delta^{23}_{max}$ and $\Delta_2 + \Delta_1 < \delta^{22}_{max}$, then the second scenario occurs (ii), and transition t_{23} is forced to fire, updating the state with the new marking $m[p_6] = 1$ and restarting the local clock for t_{22}.

In both scenarios, the enabled transitions are not enabled anymore due to a change in the marking that enabled those transitions, and then the local clock restarts. Moreover, if the transitions are enabled, and another transition fired, e.g., considering also a token in $m[p_7] = 1$ and firing the transition t_{17}, then transitions t_{22}, t_{23} are further enabled as long as the time is not greater than the upper bound, case in which a transition is forced to fire. If a robot should stop in a cell, meaning to not force a transition to fire from the respective place where a token is present, then the upper bound of the output transitions should be equal with ∞. This modification allows for the robot to not be required to depart from its current cell.

Remark 2.3 This model differs from another temporized Petri net representation, particularly Timed Petri net [83, 84], where the time expresses a deterministic value associated with the transition. Thus, a token is consumed in a fixed time instead of a time interval, as explained previously.

The TPN model serves two objectives, as follows:

1. Representation of the robotic team: by maintaining the same idea as in Chapter2.2.1, the notation $\mathcal{N}$ defines the topology of the discrete event system of the workspace for the robotic team, considering the weighted arcs to be unitary. Moreover, the marking symbolizes the number of robots in a place $p \in P$. The novelty of this model is captured in the function I that expresses the time interval for a robot to cross from one cell to an adjacent cell.

2. Representation of the robotic mission: the space constraints that should be guaranteed by the robotic system in a given time interval, can be translated into a TPN model. This approach is covered in the following section, where the purpose of the labeling function Λ will be elaborated.

Example 2.2.5 *Figure 2.8 illustrates a Time Petri net model associated with a fragment of the partitioned workspace mentioned in Example 2.2.5, directed towards cells c_7, c_8 and c_9. The addition of this representation ensures a time interval of reaching a cell from an adjacent cell. For example, the token (robot) initially placed in p_9, moves to place p_8 by firing the transition t_{22}. The time interval $[\delta^{22}_{min}, \delta^{22}_{max}]$ describes the minimum and the maximum motion time of the robot, by considering for example, the maximum and the minimum speed based on its dynamic. If the robot should wait an unlimited or unknown time in its current cell, then the upper bound changes to $\delta^{22}_{max} = \infty$. The entry into a cell from the adjacent preceding one is managed by a control procedure that can be defined by the user.* ■

Since this chapter is responsible for introducing the fundamental notions required throughout the thesis, let us define a labeling function (necessary for Chapter 5) $\Lambda : T \to \mathscr{B}'_\varepsilon$ that assigns each transition $t \in T$ a label from the alphabet set $\mathscr{B}'_\varepsilon = \mathscr{B} \cup \{\varepsilon\}$. The label ε may be repeated, while the other labels are unique to each $t \in T$. Note that the time is considered throughout the thesis in the set $\mathbb{Q}_+$.

2.3 Agents' missions

The origins of mobile robot path planning focused on exploring various techniques for proceeding from an initial point to a designated destination, primarily in single-robot scenarios [15]. Over time, the goal of reaching multiple points of interest has expanded to include scenarios where a group of agents collaborates on missions that require visiting a set of regions of interest sequentially and/or synchronously. In addition to the visiting constraints, the agents should also avoid the regions that could express obstacles. Real-life applications of autonomous agents [85] present various situations where are required to reach synchronously in a region, e.g., monitoring an area or picking up a package by multiple robots; or sequentially reaching a set of regions, e.g., picking up and delivery a package.

A natural translation of reaching and/or avoiding a set of regions of interest $\mathscr{Y}$ from a natural language to a systematic formulation that can be further interpreted and analyzed is through Boolean operators: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\Rightarrow$ (implication), and $\Leftrightarrow$ (equivalence). For example, let us consider that the multi-agent system evolves in a workspace with three regions of interest: y_1, y_2, and y_3. For all these regions, a set of atomic proposition $\mathscr{B}$ is defined, with b_1 associated with y_1, b_2 to y_2, respectively b_3 to y_3. If the team of robots should reach either region y_1, either region y_2, and avoid region y_3, then the mission can be expressed through the set $\mathscr{B}$ as $(b_1 \vee b_2) \wedge \neg b_3$. Note that the specifications are under the set of atomic propositions $\mathscr{B}$. The avoidance and reachability properties should be ensured in the final state of the robotic system. Additional characteristics can indicate if there is the need to reach/avoid a region during the trajectory (through capital letters, e.g., B_3) or at the end of the trajectory (through lowercase letters, e.g., b_1) [18]. The mission is accomplished only if the entire Boolean expression is evaluated as *True*.

The global Boolean-based formula (the mission assigned towards the entire robotic system) is expressed as a Conjunctive Normal Form (CNF) [86], by $\varphi = \varphi_i \wedge \cdots \wedge \varphi_n$, where each term φ_i is a disjunction of terms e.g., formula $\varphi = b_1 \wedge \neg b_2$ indicates the visit of region y_1 and the avoidance of region y_2.

The high-level languages defined in this section allow for specifying complex spatial and temporal constraints for the team of agents, that cannot be covered only by the use of Boolean operators.

2.3.1 Linear Temporal Logic

Bringing back to the workspace illustrated in Figure 2.8(a), let us consider the following scenario: the agents should connect to the hydrant (associated with the place p_7) in order to extinguish the fire (associated with the place p_3) while avoiding the obstacles represented by the trees (places p_2, p_5). In other words, the robotic team should first visit the region y_2, implying that immediately after the team should reach region y_3, avoiding the region y_3 all the time. Note that the objective here is to convey the order of reaching these regions of interest, which cannot be done only by using Boolean operators. Currently, the interpretation of the mission is not in focus, since critical constraints should be further considered, such as ensuring that the same robot that reached the hydrant should move afterward to the fire. In the case of a robotic system, where the robots cooperate to fulfill a mission, it may be possible for one robot to go to the hydrant and another one to go to the firing region.

Rich language expressions involving actions that are sequential or synchronous are enhanced by the addition of another set of symbols, denoted *temporal operators*, such as: until $\mathscr{U}$, eventually $\diamondsuit$, always $\square$, and next $\bigcirc$. This high-level language is known as Linear Temporal Logic (LTL) [26, 87], which are recursively defined over a set of atomic propositions $\mathscr{B}$.

Definition 2.3.1 *The syntax of LTL specifications over the set of atomic propositions $\mathscr{B}$ is defined as follows [88]:*

$$\varphi := b_k \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathscr{U}_I \varphi_2, \tag{2.2}$$

with $b_k \in \mathscr{B}$. This formalism allows also to use Boolean operators such as conjunctions $\wedge$, implication $=>$ and equivalence
$<=>$, since these operators can be defined with negation $\neg$ and disjunction $\vee$.

The tuple (r,i) denotes the formula φ which is satisfied for the run r and state i, expressed as $(r,i) \models \varphi$. Considering $b_k \in \mathscr{B}$ an atomic proposition, and two LTL formulae φ_1, φ_2, then the semantics can be recursively defined as follows:

$$(r,i) \models b_k \Leftrightarrow b_k \models r$$
$$(r,i) \models \neg\varphi \Leftrightarrow (r,i) \not\models \varphi$$
$$(r,i) \models \varphi_1 \vee \varphi_2 \Leftrightarrow (r,i) \models \varphi_1 \; or \; (r,i) \models \varphi_2$$
$$(r,i) \models \varphi_1 \mathscr{U} \varphi_2 \Leftrightarrow \exists j \geq i, s.t. (r,j) \models \varphi_2, \; and \; \forall i,k \leq i < j : (r,k) \models \varphi_1$$

The full class of LTL includes also the next operator $\bigcirc$ in the set of temporal operators. Since this operator is not suitable in the context of the discrete event system modeling the continuous robotic system [89, 90, 87], the LTL specifications used throughout the thesis are part of a subclass, as defined in Definition 2.3.1.

LTL specifications can be modeled as a discrete event system, namely a Streett or a Rabin automaton [19, 91] or a Büchi automaton [4]. In this thesis, it is considered a translation of the LTL mission into a non-deterministic Büchi automaton [92]. A deeper introduction of the procedure that associates this automaton with any LTL formula is presented in [92].

Definition 2.3.2 *The Büchi automaton for an LTL formula over the set $\mathscr{B}$ is defined as $B = (S, S_0, \Sigma_B, \rightarrow_B, F)$, where:*

- *S is a finite set of states;*
- *$S_0 \subseteq S$ is the set of initial states;*
- *Σ_B is the finite set of inputs;*
- *$\rightarrow_B \subseteq S \times \Sigma_B \times S$ is the transition relation;*
- *$F \subseteq S$ is the set of final states.* ■

As B is non-deterministic, it allows multiple transitions from a single state with the same input, e.g., $(s, \tau, s') \in \rightarrow_B$ and $(s, \tau, s'') \in \rightarrow_B$, with $s' \neq s''$. Thus, an input sequence can produce more than one sequence of states. The set of inputs that enable a transition from s_i to s_j, denoted by $\pi(s_i, s_j)$, is expressed as a Boolean formula over $\mathscr{B}$ in Disjunctive Normal Form (DNF). The inputs can also be represented as a combination of active observations over the power set $2^{\mathscr{B}}$. Through active observation, the *True* value of an atomic proposition $b \in \mathscr{B}$ is understood.

An infinite input word (a sequence with elements from Σ_B) is *accepted* by B if it generates at least one sequence of states (referred to as a *run*) in B that visits the set F infinitely often. Concomitantly, a run of B that infinitely often visits set F is called *accepted*, and if B has at least one accepted (infinite) run that it has at least one run with a finite prefix-suffix representation [92]. This means that the accepted run (and the corresponding input word) can be stored on finite memory in terms of two finite-length strings: (i) *prefix* - leading to a final state in set F, and (ii) *suffix* - returning to the same final state reached by the

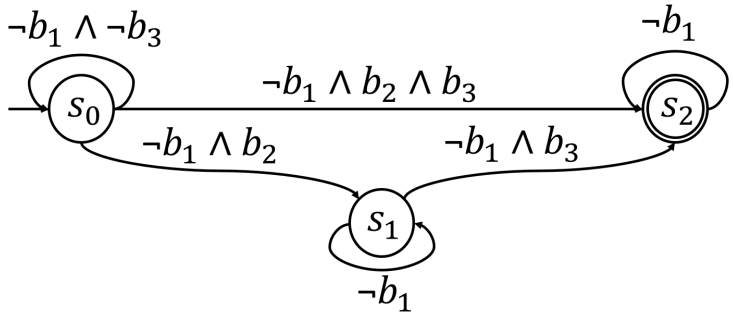


Fig. 2.9. Büchi automaton for the LTL formula in equation (2.3)

prefix. The run of B is formed by the prefix followed by infinite repetitions of the suffix, i.e. it is written as *prefix, suffix, suffix,* ... using the sequence of elements from Σ_B. Automatic translation from an LTL formula to a Büchi automaton can be achieved using various tools, including those discussed in [21, 93, 94].

Considering the motion planning field, where the robots should fulfill a mission related to the environment, particularly to reach and/or avoid a set of regions of interest, the set of atomic propositions $\mathscr{B}$ is directly related to the set $\mathscr{Y}$. An input in the Büchi automaton associated with a transition from one state to an adjacent one is expressed through a Boolean formula, e.g., $b_i \wedge b_j$, with $b_i, b_j \in \mathscr{B}$. This formula translates afterward into an action that the multi-robot system is required to ensure regarding the mission. Specifically, b_i represents the atomic proposition for region y_i, being *True* when the region y_i is visited, respectively b_j represents the region y_j. Therefore, the transition in the automaton is enabled only when the robots reach simultaneously the regions of interest y_i, y_j.

The relation between the robotic RMPN model and the Büchi automaton representation is established through the set $\mathscr{B}$, connecting the movements of the robots with the LTL mission. Later on, in Chapter 4.2, a translation between the automaton and a Petri net model is investigated, taking into account the relevance of set $\mathscr{B}$.

Example 2.3.3 *As mentioned previously, the mission that is of interest for Figure 2.2(a) specifies to "Eventually reach region y_2 (the hydrant), as well as y_3 (the fire), yet y_2 should be visited before y_3 and always avoiding the obstacles y_1 (the trees)". This mission expressed in the LTL formalism can be written as in equation* (2.3)*, based on the set $\mathscr{B}$.*

$$\varphi = \Diamond b_2 \wedge \Diamond b_3 \wedge \neg b_3 \mathscr{U} b_2 \wedge \Box \neg b_1 \tag{2.3}$$

The Büchi automaton for this LTL mission is visualized in Figure 2.9. The initial state is s_0 and the final state is represented by s_2 (emphasized by the double edge). One possibility to fulfill the mission is to reach the final state s_2 through the accepted run $s_0, s_1, s_2, s_2, \ldots$ with the prefix s_0, s_1 and the suffix s_2. The avoidance of the region y_1 is observed throughout the

entire state space of the Büchi automaton. As long as b_1 (related to y_1) is evaluated as False, then the final state is visited infinitely often.

The mentioned run is not unique, as the state s_1 can be visited multiple times, e.g., the prefix is $s_0, s_1, s_1, \ldots$. Another run for ensuring the mission φ is represented by the sequence $s_0, s_2, s_2, \ldots$. Notice that this run can only be achieved if the robotic team consists of a minimum of two robots, as both regions should be visited simultaneously y_2 and y_3.

Although this run satisfies the LTL formula, it does not solve the request of reaching first the hydrant before reaching the area with the fire. This occurs since the formula is verified from a logical point of view. This scenario can be avoided if the formula includes also the expression $\square\neg(b_2 \wedge b_3)$. Remark that even if this modification over the formula is made, there is no requirement that the same agent reaches the region y_2 before reaching region y_3. This hindrance can be resolved either by defining a set of atomic propositions suitable for the problem rather than associating the set $\mathscr{B}$ with the set $\mathscr{Y}$, either by adding constraints when applying the motion planning strategy for the robotic team and the given LTL specification. ■

2.3.2 Metric Interval Temporal Logic

In contrast with the Linear Temporal Logic that embodies spatial and temporal constraints for the multi-agent system, in terms of synchronization and sequencing of the regions of interest, the Metric Interval Temporal Logic (MITL) provides time-related constraints expressing a time interval in which a region should be reached.

Definition 2.3.4 *The syntax of MITL specifications over the set of atomic propositions $\mathscr{B}$ is defined as follows [95]:*

$$\varphi := b_k \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc_I \varphi \mid \Diamond_I \varphi \mid \square_I \varphi \mid \varphi_1 \mathscr{U}_I \varphi_2, \tag{2.4}$$

where $b_k \in \mathscr{B}$, I is a non-empty time interval $[i_1, i_2]$ or (i_1, i_2), with $i_1 < i_2$, $i_1 \in \mathbb{N}$, and $i_2 \in \mathbb{N} \cup \{\infty\}$. The intervals $[0, i_2]$ and $[0, i_2)$ are denoted by $\leq i_2$ and $< i_2$, respectively. MITL uses Boolean operators such as negation $\neg$, and $\wedge$, as well as temporal operators such as next $\bigcirc$, eventually $\Diamond$, always $\square$, and until $\mathscr{U}$.

The tuple (w^t, i) *defines the MITL formula* φ *over* $\mathscr{B}$ *with the timed word* $w^t = (w_0, \tau_0)(w_1, \tau_1)\ldots$ *and is recursively satisfied as follows:*

$$
\begin{aligned}
&(w^t, i) \models b_k \Leftrightarrow b_k \in w_i \\
&(w^t, i) \models \neg\varphi \Leftrightarrow (w^t, i) \not\models \varphi \\
&(w^t, i) \models \varphi_1 \wedge \varphi_2 \Leftrightarrow (w^t, i) \models \varphi_1 \textit{ and } (w^t, i) \models \varphi_2 \\
&(w^t, i) \models \bigcirc_I \varphi \Leftrightarrow (w^t, i+1) \models \varphi \textit{ and } \tau_{i+1} - \tau_i \in I \\
&(w^t, i) \models \Diamond_I \varphi \Leftrightarrow \exists j \geq i, s.t. (w^t, j) \models \varphi, \tau_j - \tau_i \in I \\
&(w^t, i) \models \Box_I \varphi \Leftrightarrow \forall j \geq i, \tau_j - \tau_i \in I \Rightarrow (w^t, j) \models \varphi \\
&(w^t, i) \models \varphi_1 \mathscr{U}_I \varphi_2 \Leftrightarrow \exists j \geq i, s.t. (w^t, j) \models \varphi_2, \\
&\qquad\qquad \tau_j - \tau_i \in I \textit{ and } (w^t, k) \models \varphi_1, \forall i \leq k < j
\end{aligned}
$$

Similar to the translation of an LTL formula into a Büchi automata, fragments of MITL formula benefit from a representation denoted Timed Büchi automata (TBA), as encountered in literature [95–97]. The chosen method employed in this thesis follows the approach presented in [95] and detailed applied in [98].

The time units used in the bounded time limit of these specifications are user-defined, e.g., seconds, or minutes. As mentioned in [99], the MITL formulae are expressible in both continuous and point-wise. For an easier understanding of the meaning of an MITL mission φ, the following missions are expressed.

- The MITL formula $\varphi = \Box_{[0,20]} \neg b_1$ specifies that region y_1 is always avoided for the first 20 time units;
- The MITL formula $\varphi = \Diamond_{[0,\infty]} b_2 \wedge \Diamond_{[2,2]} b_3$ requires the visit of region y_2 at any time and the visit of region y_3 after exactly 2 time units.

Let $X = \{x_1, x_2, \ldots, x_{|X|}\}$ denote a finite set of clocks. this notion is required to be explained to understand the Timed Büchi automata definition. The set of *clock constraints*, $\Phi(X)$, is defined by the following grammar [72]:

$$\phi := \top \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid x \bowtie \psi, \tag{2.5}$$

where $x \in X$ is a clock, $\psi \in \mathbb{Q}_+$ is a constant, and $\bowtie \in \{<, >, \leq, \geq, =\}$. A *clock valuation* ν is a function $\nu : X \to \mathbb{Q}_+$ assigning a non-negative rational value to each clock. The operation $\nu + \delta$ increments every clock by $\delta \in \mathbb{N}_+$, i.e., $(\nu + \delta)(x) = \nu(x) + \delta$. The satisfaction of the clock constraint ϕ by the valuation ν is denoted by $\nu \models \phi$.

The Timed Büchi Automaton model adheres to the definition in [72], following notations from [100], with the note that this work does not distinguish between final and repeated locations. Moreover, these notations are relevant in Chapter 4, where a method of path planning for a robotic team requires the translation from a TBA model to a Time Petri net one. Note that the automata used in this thesis is considered to be deterministic.

Definition 2.3.5 *A Timed Büchi Automaton (TBA)* $\mathscr{A} = \langle Q, q_0, X, \Phi(X), \Sigma, E, Inv, F \rangle$ *is defined as follows:*

- Q *is the finite set of locations;*
- $q_0 \in Q$ *represents the initial location;*
- X *if the finite set of clocks;*
- $\Phi(X)$ *represents the clock constraints;*
- $Inv : Q \to \Phi(X)$*: is the invariant function;*
- Σ *denotes the alphabet set;*
- $E \subseteq Q \times \Phi(X) \times \Sigma \times 2^X \times Q$ *is the set of edges, where an edge* $e = (q, \gamma, \lambda, R, q')$ *has guard* $\gamma \in \Phi(X)$*, label* $\lambda \in \Sigma$*, and reset set* $R \subseteq X$*;*
- $F \subseteq Q$ *is the set of accepting locations.*

A state of the automata $\mathscr{A}$ is a pair (q, v) with $q \in Q$ and $v \models Inv(q)$. The initial state is $(q_0, 0)$, where 0 maps every clock to 0. There are two types of transitions: (i) *discrete transition* $(q, v) \xrightarrow{e} (q', v')$ exists if $v \models \gamma$ and $v' \models Inv(q')$, with the clocks in R are reset to 0, while others remain unchanged; (ii) *time transition* $(q, v) \xrightarrow{\delta} (q', v')$ for $\delta \in \mathbb{Q}_+$ exists if $q = q'$, $v' = v + \delta$, and $v' \models Inv(q)$. Due to the time-additivity property [101], $(q, v) \xrightarrow{\delta} (q', v')$ and $(q', v') \xrightarrow{e} (q'', v'')$ can be combined as $(q, v) \xrightarrow{\delta} \xrightarrow{e} (q'', v'')$.

Clock reset is not mandatory for a single instance, for example using one temporal operator *eventually* $\Diamond$. However, clock reset becomes essential in nested MITL specifications [102], where the temporal operators are nested and the time interval of one operator depends on the previous one, e.g., $\Box_{[0,7]}(\Diamond_{[1,2]} b_1 \to \Diamond_{[3,6]} b_2)$ requiring that in the time interval [0,7], if region y_1 is visited within [1,2], then the region y_2 should be visited within [3,6] time units. The work presented in this thesis do not make use of nested MITL operators.

The satisfaction of an MITL formula is similar to the satisfaction of an LTL formula: by finding an *infinite accepted run* in the associated automata. In the TBA $\mathscr{A}$, this run is a sequence of time and discrete transitions $(q_0, v_0) \xrightarrow{\delta_0} (q'_0, v'_0) \xrightarrow{e_0} (q_1, v_1) \xrightarrow{\delta_1} (q'_1, v'_1) \ldots$ that starts from an initial state and reaches a final one in F. The final state should be visited infinitely often.

Example 2.3.6 *This example illustrates the modality of expressing the sequential visit of the robotic team for regions* y_2 *(hydrant) and* y_3 *(fire) from Figure 2.2(a), as an MITL formula. Specifically, the equation* (2.6) *states that the visit of the region* y_2 *should be reached in 5-time units, and this visit implies an eventual visit of the region* y_3 *in the next 10 time units.*

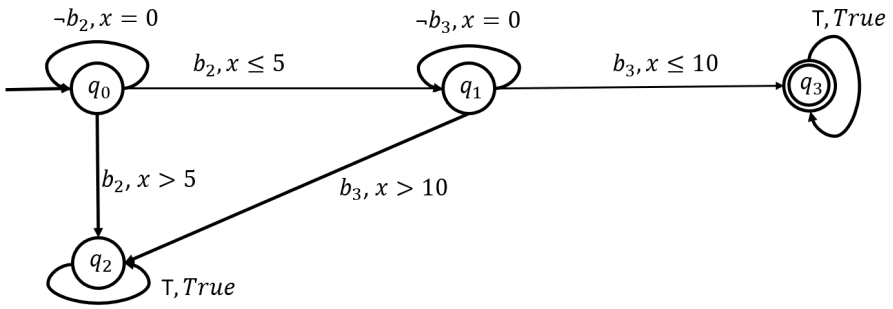


Fig. 2.10. Timed Büchi automaton for the MITL formula in equation (2.6)

$$\varphi = \Diamond_{[0,5]} b_2 \wedge (b_2 \rightarrow \Diamond_{[0,10]} b_3) \tag{2.6}$$

Figure 2.10 portrays the TBA model associated with the MITL mission previously described, based on the algorithm explained in [98]. This modeling is based on the fact that the conjunction operator is not commutative, in the sense that the clock of a temporal operator resets with respect to the first temporal operator. The automata contains 4 states, with the initial q_0 and final q_3 (visualized through double edge) states. The states q_2 and q_3 are considered to be sink states [103] denoting either an error, respectively an accepting state, from which the run sequence remains in that state [104]. The continuous loop is observed by the output arc from these states that is evaluated as True (T), for any time.

The edges between the states include both space and time requirements, such that the following reached state respects the imposed constraints, e.g., the state q_1 is reached only if the atomic proposition y_2 is true while the clock constraint is less than the upper bound 5; otherwise, the error state q_2 can be reached if the clock does not act according with the imposed limit of 5-time units. Throughout the representation of the automata A, each state resets the clock $x = 0$. Moreover, there is no need of using multiple clocks, since no nested formulae are part of the MITL mission. ■

Remark 2.4. The benefit of adopting a set of actions of the robots as the set of atomic propositions is captured in Chapters 4 and 7, for specifications under MITL formalism.

2.4 Comparison criteria for planning strategies of multi-agent systems

This section covers the chosen metrics considered for in the evaluation comparison between the proposed methods and other DES planning solutions. Their purpose is to assess the quality of the results through a comparison analysis procedure.

- **(a) Model size**. This metric is computed in two scenarios depending on the methods that are compared: the model size includes all the places and transitions of a composed model under the Petri net representation, capturing both the robotic model and the given specification [35]; the model size includes only the places and transitions of the robotic model under the Transition System representation when the methods have a sequential approach for the models of the team and the mission [4].
- **(b) Run time** to return a solution such that a given LTL mission is ensured by the robotic team. This metric excludes the time needed to build the model.
- **(c) Trajectory length** for the whole robotic team obtained, expressed as the total number of fired transitions in the robotic model. This metric is associated with the number of cells that are being crossed by the robotic team, in the partitioned workspace.

Two planning solutions for robotic teams ensuring their global mission are described as follows. These methods serve as comparison approaches when validating the proposed techniques as part of this thesis.

(i) **FB [4]** - this method (*following Büchi (WB)*) aims for a sequential approach of computing trajectories of the robotic team by following a run (a path from the initial state towards one final state) in the Büchi automaton. First, the PN model is assigned to the team, based on the partitioned workspace. Secondly, an optimization MILP problem is solved in the search for a sequence of markings that can generate the necessary observations that fulfill the LTL mission based on a set of feasible runs computed in the Büchi automaton. The approach is iterative until the team can act under the accepted run in the automaton, producing a sub-optimal solution that cannot ensure collision-free trajectories.

(ii) **TS [50]** - this approach (*Transition Systems (TS)*) is subject to represent the motion of each robot as a Transition System model versus the previous approaches where one single model is assigned for the entire robotic team based on a product automata procedure. Thus, the state of the entire robotic team is provided through this single model. In addition to this, the automaton modeling the mission is also joined with the robotic model. Robots' trajectories are computed by a graph-search-based algorithm for the built-composed model.

The **FB** method is based on a sequential approach of using two representations associated with the robotic team and the given LTL mission. Since the following chapter introduces a framework directed towards a parallel use of these two representations, the detailed description of the **FB** method is presented in Chapter4.1.

Chapter 3

Task decomposition approach for multi-agent systems

This chapter presents an approach for task allocation in a multi-robotic system, where both path planning and collision avoidance problems are taken into account. The proposed method involves task decomposition for a global co-safe linear temporal logic (LTL) specification, where independent tasks are assigned to each robot. The results of this method are evaluated in a 3D workspace which is partitioned into a set of cells returned by a personal 3D cell decomposition technique. The simulations include the motion of a team of UAVs ensuring the global LTL mission.

The main contribution of this chapter is represented by a solution for task decomposition of global formalism expressed as co-safe LTL specifications, leading to independent trajectories for each robot without the need to communicate or synchronize between them. Decomposing tasks in a multi-robot system is crucial when addressing complex global missions, as it allows for a scalable and efficient solution that leverages the capabilities of each robot. A global mission, typically specified for the entire team, can often be too complex or computationally demanding for centralized approaches, particularly as the number of agents increases. By breaking down the mission into smaller, independent tasks, each robot can operate autonomously, reducing the need for continuous communication and synchronization. This decomposition is especially valuable when the mission is expressed using the Linear Temporal Logic (LTL) formalism, which enables the specification of intricate temporal behaviors, such as ordered sequences of tasks, or synchronization, as defined in Section 2.3.1.

In decentralized systems, task decomposition ensures that each robot's trajectory can be synthesized to locally satisfy parts of the global LTL formula, reducing computational overhead [43]. Moreover, independent task execution mitigates the risk of communication failures, a common issue in centralized systems, while still guaranteeing the global mission's correctness through the satisfaction of the co-safe LTL specifications [105, 106]. Thus,

decomposing tasks not only simplifies mission execution but also enhances the system's robustness and scalability.

This approach focuses on developing a planning strategy for a multi-robot system based on a high-level specification in the formalism of Linear Temporal Logic (LTL). The proposed solution provides the foundation for future path-following controllers, to distribute tasks encapsulated in a global mission. By decomposing the global task into independent tasks, the need for intermediate coordination or synchronization among robots is eliminated, except for the application of local collision-avoidance rules when robots are in close proximity. The primary objective is to generate *independent* trajectories for each robot, ensuring that their collective movement satisfies the specification given for the entire team. The term "independent" implies that each robot can execute its motion without requiring communication or synchronization with other robots, distinguishing this approach from centralized methods like those presented in [107]. The global specification is expressed as a co-safe LTL_X formula over a set of regions of interest $\mathscr{Y}$.

In this context, each robot is modeled as Transition System (TS). This model abstracts the robot's potential motions within the environment E, enabling seamless integration with the Büchi automaton corresponding to the LTL formula over the set of atomic propositions $\mathscr{B}$ related to the set $\mathscr{Y}$.

Example 3.0.1 *Starting with this example, other illustrative ones are included throughout the thesis, to provide a clearer image of the theoretical notions. In Figure 3.1(a) there is represented a 3D environment E including four disjoint regions of interest. The initial location of the two robots is highlighted with colors red (for r_1) and blue (for r_2) circles. The imposed specification is*

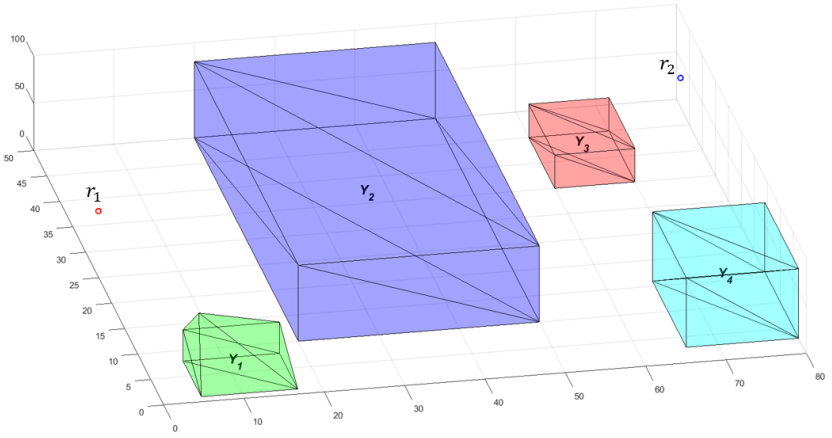
$$\varphi = \Diamond b_1 \wedge \neg b_2 \mathscr{U} (b_3 \vee b_4), \tag{3.1}$$

requiring for (1) region y_1 to be eventually reached, while (2) region y_2 is avoided until of the y_3 and y_4 regions is eventually reached. Intuitively, if a robot ensures part (1) while it avoids y_2 and the other robot ensures part (2) of φ, then the robots can do this in a distributed manner, for example without any synchronization or communication. On the contrary, if the last parenthesis is changed to $(b_3 \wedge b_4)$, then both robots are needed to cooperate in satisfying the mission. One situation is if one robot reaches y_3, it should wait until the second robot visits the disjoint region y_4.

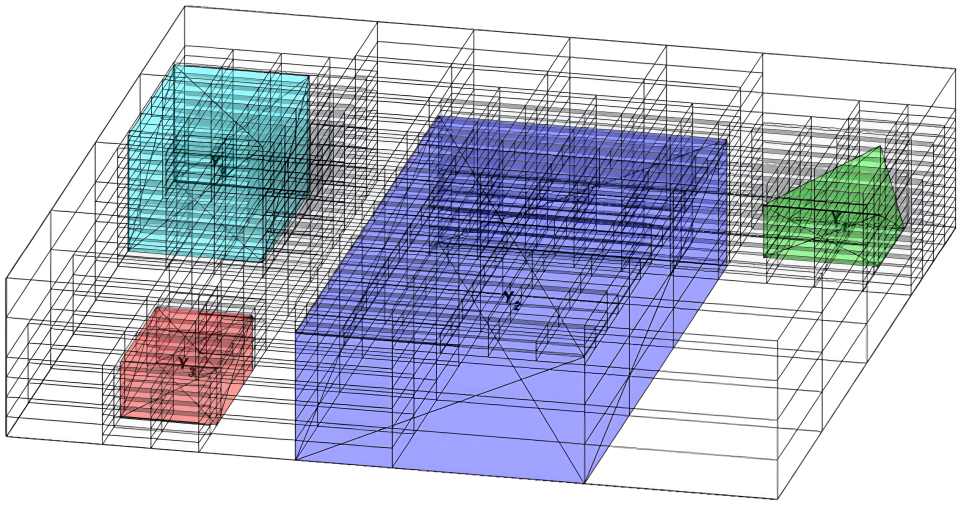
Figure 3.1(b) reveals a visual representation of the partitioning of the environment, with precision $\varepsilon = 16$. ■

3.1 Task decomposition

To distribute a given LTL mission φ among robots in a team, this section focuses on automatically decomposing φ into a set of sub-formulas, referred to as *tasks*, which satisfy certain



(a) Environment with four regions of interest and initial positions of two robots (r_1 - red, r_2 - blue)



(b) 3D decomposition in rectangular cuboids considering a precision $\varepsilon = 16$

Fig. 3.1. 3D workspace with four regions of interest and the discrete representation of the partitioned environment.

properties. Each task is designed to be executed independently by a single robot, and the decomposition must meet two key requirements: *independence* and *fullness*. *Independence* implies that tasks are non-conflicting and can be accomplished without synchronization between robots. Specifically, a task φ_i in the decomposition $\{\varphi_1, \ldots, \varphi_{|\mathscr{R}|}\}$ should not interfere with another task φ_j, where $j \neq i$. *Fullness* ensures that the global mission φ is achieved when all tasks are completed.

The proposed method, outlined in Algorithm 2, begins by converting the LTL formula φ into a Büchi automaton B using existing tools from [93]. In addition to B, the algorithm requires two inputs: a set $\mathscr{C}$ representing the partitioned environment and an observation map h, which associates partition cells with regions of interest $\mathscr{Y}$. For simplicity, the set O_S (line 1 of the algorithm) is used to represent all observations from the atomic proposition set $\mathscr{B}$, corresponding to regions in $\mathscr{Y}$, that can be generated by a single robot.

The algorithm starts by trimming the Büchi automaton (lines 1–3) to ensure that all transitions in B can be enabled by a single robot. This trimming step is adapted from [107], which originally addressed cooperative robots, to suit independent task execution. The goal is to decompose φ into tasks associated with elements of Σ_B (the input set of B), guaranteeing that each task can be performed by a single robot. Without trimming, certain transitions might require collaboration between robots.

Next, all loopless accepted runs of B are computed (lines 4–11) using the k-shortest path algorithm [108]. For each pair of initial and final states (s_0, s_f), paths are iteratively computed, increasing k as needed, until all loopless paths are included. The resulting set *Runs* contains all such accepted runs.

The decomposition process continues by selecting a run $\rho \in Runs$ (line 13) and verifying whether all permutations of its transitions correspond to valid runs in B. If this condition is satisfied, the transitions of ρ define a decomposition set $Decomp_\rho$, where each task corresponds to an observation set realizable by a single robot. These tasks ensure the fullness property, as ρ represents an accepted run of B.

To ensure independence, the algorithm processes self-loops of states in B (lines 23–25). Self-loops with the same output transitions are replaced by their intersection, preventing any task from violating the global mission φ. Finally, the algorithm removes ρ and its permutations from *Runs* (lines 14 and 21) to avoid redundant iterations.

By iteratively applying this procedure, the algorithm generates a decomposition of φ into independent and complete tasks that can be executed by individual robots, achieving the global mission.

Algorithm 2 involves a significant number of iterations, as suggested by its pseudo-code. The trimming of the Büchi automaton has a linear complexity with respect to the number of transitions in $\rightarrow_B$, while the size of B is determined by the imposed LTL specification. The k-shortest path algorithm is repeatedly executed in lines 5–11 for an initially unknown

Algorithm 2: Mission decomposition

Input : Büchi automaton B, partition $\mathscr{C}$ and observation map h
Output : Feasible decompositions *TaskSet*

Compute $O_S = \bigcup_{c \in \mathscr{C}} h(c)$
for $\left(s_i, \rho(s_i, s_j), s_j\right) \in \rightarrow_B$ **do**
  $\rho(s_i, s_j) = \rho(s_i, s_j) \cap O_S$
  Initialize $Runs = \emptyset$
  for $s_0 \in S_0$ *and* $s_f \in F$ **do**
    $k = 0$
    repeat
      $k = k + |S|$
      Let $Runs_{s_0 \to s_f} = k\_shortest\_path(k, s_0, s_f)$
    until $|Runs_{s_0 \to s_f}| < k$;
    $Runs := Runs \cup Runs_{s_0 \to s_f}$
while $Runs \neq \emptyset$ **do**
  Choose a run $\rho \in Runs$
  $Runs = Runs \setminus \{\rho\}$
  Compute $Decomp_\rho = \bigcup_{i=1}^{|\rho|-1} \{\rho_B(\rho(i), \rho(i+1))\}$
  Set $counter = 1$
  for $\gamma \in Runs$ **do**
    if $(|\gamma| = |\rho|)$ **then**
      Compute $Decomp_\gamma = \bigcup_{i=1}^{|\gamma|-1} \{\rho_B(\gamma(i), \gamma(i+1))\}$
      if $\left((Decomp_\rho \setminus Decomp_\gamma) = \emptyset \right)$ **then**
        $Runs = Runs \setminus \{\gamma\}$
        $counter = counter + 1$
        for $\rho(i) \in \rho$ *and* $\gamma(j) \in \gamma$ *such that sets* $\rho_B(\rho(i), \rho(i+1))$ *and* $\rho_B(\gamma(j), \gamma(j+1))$ *are identical* **do**
          Set $\rho_B(\rho(i), \rho(i)) = \rho_B(\rho(i), \rho(i)) \cap \rho_B(\gamma(j), \gamma(j))$
          Set $\rho_B(\gamma(j), \gamma(j)) = \rho_B(\rho(i), \rho(i)) \cap \rho_B(\gamma(j), \gamma(j))$
        if $\left(counter = (|\rho|!) \right)$ **then**
          $Decomp_\rho$ is a feasible decomposition; append it to *TaskSet*
if *TaskSet is empty* **then**
  Mission cannot be decomposed

number of iterations, dictated by the structure of B. However, each of these executions has a pseudo-polynomial complexity [108].

While lines 12 and 17 impose up to $|Runs|^2$ iterations, these involve relatively simple set operations, and the cardinality of *Runs* is significantly reduced by trimming B, which simplifies the automaton's structure. This trimming step is essential not only for ensuring task independence but also for reducing computational overhead. Importantly, the complexity of Algorithm 2 is independent of the number of robots $|\mathscr{R}|$.

However, the algorithm is not complete in terms of generating all possible decomposition sets. It does not consider sequences of multiple tasks that could be assigned to a single agent while still ensuring the independence and fullness properties.

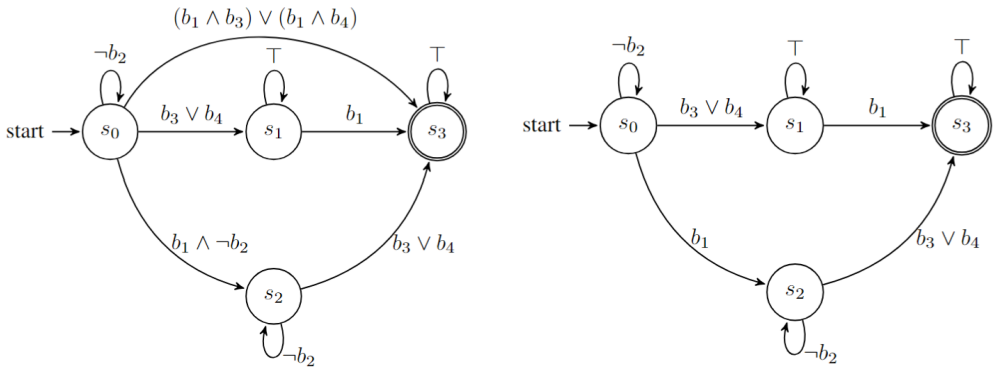
Example 3.1.1 *Consider the LTL formula from Equation* (3.1)*,* $\varphi = \diamondsuit b_1 \wedge \neg b_2 \mathscr{U} (b_3 \vee b_4)$*. The Büchi automaton corresponding to this formula is shown in Figure 3.2(a). The mission described by* φ *requires that region* y_1 *is eventually reached while region* y_2 *is avoided until one of the regions* y_3 *or* y_4 *is eventually visited. Intuitively, this mission can be distributed among robots without requiring processes such as synchronization or communication. For example, one robot can reach* y_1 *while avoiding* y_2*, and another can fulfill the second part of the formula.*

However, if the last parenthesis in φ *is modified to* $(b_3 \wedge b_4)$*, implying both regions* y_3 *and* y_4 *must be visited simultaneously, cooperation would be required. In this case, a robot reaching* y_3 *would need to wait until another robot visits the disjoint region* y_4*.*

Before applying the task decomposition algorithm, the Büchi automaton B needs to be trimmed. Observations such as $b_1 \wedge b_3$ *or* $b_1 \wedge b_4$ *are redundant because regions* y_1*,* y_3*, and* y_4 *are disjoint, meaning a robot cannot simultaneously occupy these regions. Similarly, the observation enabling the transition from* s_0 *to* s_2 *becomes* b_1*, as regions* y_1 *and* y_2 *are also disjoint. Figure 3.2(b) illustrates the trimmed Büchi automaton obtained after executing lines 1–3 of Algorithm 2. The resulting set of accepted runs after line 11 is* $Runs = \{\rho_1 = s_0, s_2, s_3; \rho_2 = s_0, s_1, s_3\}$*. Considering the implementation process, constructing B, trimming it, and computing the set Runs is less than 0.22 seconds.*

During the first iteration of Algorithm 2, ρ_1 *is selected (line 13), yielding the decomposition* $Decomp_{\rho_1} = \{\{b_1\}, \{b_3 \vee b_4\}\}$ *(line 15). After* ρ_1 *is removed, only* ρ_2 *remains for the iteration starting at line 17. For* ρ_2*, the decomposition* $Decomp_{\rho_2} = \{\{b_3 \vee b_4\}, \{b_1\}\}$ *is computed (line 19). Since the decomposition sets are identical (line 20), the condition in line 26 is satisfied, and the final decomposition outputs* $\{\{b_1\}, \{b_3 \vee b_4\}\}$*.*

To ensure that tasks do not violate the mission during independent execution, the self-loops of states are adjusted as described earlier. For example, the self-loop of s_1 *is updated to* $\neg b_2$*, similar to that of* s_2*, ensuring that region* y_2 *is not visited in any trajectory. In this simple example, Algorithm 2 returned the formula decomposition in approximately 0.04 seconds.* ■



(a) Original Büchi automaton (b) Trimmed Büchi automaton

Fig. 3.2. Büchi automaton corresponding to the LTL formula $\varphi = \Diamond b_1 \wedge \neg b_2 \mathscr{U} (b_3 \vee b_4)$ (a) and the trimmed automaton after running Algorithm 2 which tailors the self-loop of s_1 to $\neg b_2$

3.2 Solution for task allocation

If Algorithm 2 returns a nonempty *TaskSet*, it indicates that the formula can be decomposed into independent tasks, which can then be assigned to the $|\mathscr{R}|$ robots. Any decomposition from *TaskSet* may be selected, denoted generically as $\bigcup_{i=1}^{|\rho|-1}\{\rho_B(\rho(i),\rho(i+1))\}$. As previously discussed, this decomposition consists of elements of Σ_B that enable transitions along the run ρ of B.

For simplicity, let us denote each task $\rho_B(\rho(i),\rho(i+1))$, where $i = 1,\ldots,|\rho|-1$, by φ_i. Each task φ_i comprises the inputs of B that trigger a transition from state $\rho(i)$ to $\rho(i+1)$. To accomplish a task φ_i, a robot must generate any observation belonging to φ_i (i.e., an element of $2^{\mathscr{B}}$).

Since B was trimmed prior to computing its accepted runs, it is guaranteed that any element of φ_i can be produced by the proper positioning of a single robot. For instance, φ_i cannot represent a conjunction of observations from two disjoint regions. Furthermore, the independence and fullness properties ensure that the original LTL formula is satisfied if all tasks φ_i are independently completed by the robots. This eliminates the need for synchronization or a specific order in visiting the regions.

Therefore, the $|\rho|-1$ tasks φ_i should be allocated to the $|\mathscr{R}|$ robots, by following the steps:

(i) Construct $W \in \mathbb{R}^{(|\rho|-1)\times|\mathscr{R}|}$, where W represents a cost matrix, with $W(i,r)$ being the cost incurred if the task φ_i is satisfied by the robot r.

(ii) Computing matrix W to assign all tasks to agents such that a desired cost function for the whole team is minimized.

For fulfilling the first step, let us iterate a procedure that drives each robot $r \in \mathscr{R}$ from initial deployment to a position that satisfies a task φ_i, $i = 1,\ldots,|\rho|-1$. This procedure

is a graph search, but instead of considering the entire set $\mathscr{C}$ which can be associated with the model of a single robot evolving in the workspace, a reduced model is considered as follows. Robot r should reach a place where where an element of $\rho_B(\rho(i),\rho(i+1))$ is evaluated as $True$ (for satisfying φ_i), hence the set of possible destination nodes for r is represented by $D = \{c \in \mathscr{C} \mid h(c) \subseteq \rho_B(\rho(i),\rho(i+1)) \text{ and } mixed(c) = 0\}$. Note that cells labeled with *mixed* are avoided since there is the need to generate the $True$ value of an atomic proposition which cannot be guaranteed by a cell including partially a region of interest. As the robot r moves toward any state in the set D, it should generate observations from the set $\rho_B(\rho(i),\rho(i))$ at every intermediate position. This ensures that the state $\rho(i)$ of B is maintained until the transition to $\rho(i+1)$ is enabled. Consequently, r is permitted to traverse only through intermediate nodes in the set $I = \{c \in \mathscr{C} \mid h(c) \subseteq \rho_B(\rho(i),\rho(i))\}$.

Thus, from the entire representation of the environment, a reduced discrete representation is obtained, having as nodes the cells from the reunion of sets: D, I, particularly $D \cup I$ and the adjacency relation between these cells. Afterward, a shortest path search algorithm is computed on the latter model, since the set of cells can be represented by a graph. Thus, a run from the initial node p_{S_0} to any node from set D can be computed. The used algorithm is Dijkstra [109], and the cost of the returned path is the value to be saved in $W(i,r)$.

The procedure detailed before is captured by lines 7-15 from the overall algorithmic solution provided in Algorithm 3.

Remark 3.1 In some cases, the graph search algorithm may fail to find a solution for a given pair (φ_i, r), specifically when c_{S_0} is not part of or is disconnected from the set $D \cup I$. This indicates that robot r cannot reach a position where φ_i evaluates as $True$ while ensuring that the observations along its path remain within $\rho_B(\rho(i),\rho(i))$. In such situations, a large value N (representing the infinite cost) is stored in $W(i,r)$.

If the resulting matrix W contains at least one row with all entries equal to N, it implies that the current task distribution is infeasible. In this case, an alternative distribution from those generated by Algorithm 2 should be chosen, or a centralized approach, as described in [107], should be applied. The condition in line 16 of Algorithm 3 accounts for such cases by switching to the centralized planning method from [107].

The following observations summarize the explanation provided earlier:

1. The graph search approach used here differs from the methods in [23, 107], where two versions of a Mixed Integer Linear Programming (MILP) problem were employed to compute $W(i,r)$. In [107], the MILP approach was necessary for centralized planning of the entire robot team using a Petri net model, while in [23], the MILP formulation was adapted for a single robot. Compared to MILP optimization, which belongs to the NP-hard class, the current approach benefits from lower computational complexity, as the Dijkstra graph search algorithm has a complexity of $O(|P_S|^2)$.

2. The costs in W are calculated as the sum of transition weights in the graph, allowing them to represent metrics like expected time or energy required for moving between

adjacent cells in the environment. In this work, unitary transition costs are assumed, minimizing the number of direction changes during robot movement across cells.

3. Each cost $W(i,r)$ is computed assuming that robot r starts from its initial position c_{S_0}. However, when multiple tasks are assigned to a single robot, the total cost incurred will differ slightly from the simple sum of the corresponding costs in W.

Step (ii) represents an optimal task allocation method, where $|\rho|-1$ tasks must be independently assigned to $|\mathscr{R}|$ agents, minimizing a cost derived from W. This step is formalized as the MILP problem in Equation (3.2) [23], with the following details:

- The decision variables are a binary matrix $Z \in \{0,1\}^{|\mathscr{R}|\times(|\rho|-1)}$ and a real variable λ.
- The solution Z specifies the tasks assigned to each robot, where $Z(r,\varphi_i)=1$ indicates that robot r is responsible for task φ_i.
- The variable λ facilitates the minimax optimization, aiming to minimize the maximum cost incurred by any robot.
- The cost function includes a term that minimizes the number of cells visited (via λ) and another term that discourages unnecessary movements of faster robots.
- Constraints ensure that all tasks from the chosen decomposition are accomplished and that the cost for each robot does not exceed λ.

$$\begin{aligned} &\min N\cdot\lambda+\textstyle\sum_{r=1}^{|\mathscr{R}|} Z(r,:)\cdot W(:,r)\\ &\text{s.t. } \textstyle\sum_{r=1}^{|\mathscr{R}|} Z(i,r)=1,\ \forall i=1,\ldots,|\rho|-1\\ &\quad Z(r,:)\cdot W(:,r)\le\lambda,\ \forall r=1,\ldots,|\mathscr{R}|\\ &\quad Z\in\{0,1\}^{|\mathscr{R}|\times|\rho|-1},\ \lambda\in\mathbb{R}_{\ge 0} \end{aligned} \tag{3.2}$$

MILP (3.2) can be solved using tools such as [110]. Its feasibility is guaranteed because the MILP is only invoked when all tasks are achievable, as stated in Remark 3.2. The solution Z specifies the task allocation but does not dictate their order. To avoid computational overhead, tasks for each robot are ordered based on their cost in W, with lower-cost tasks prioritized (line 22).

The pseudo-code for the entire method is presented in Algorithm 3, building the trajectories (sequences of cells) to be individually followed by each robot. The approach assumes that an individual robot cannot solve the problem in two scenarios: when the mission cannot be decomposed or when the selected decomposition is infeasible, as mentioned in the previous remark. In either case, the centralized method from [107] can be employed to generate a solution, requiring the robots to communicate and synchronize along their paths (lines 5 and 17 of Algorithm 3).

If a feasible decomposition is identified, MILP (3.2) is utilized to assign individual tasks to robots. For each robot r, the sequence of cells Seq_r is constructed such that the robot sequentially completes its assigned tasks in ascending order of expected cost from matrix W (determined on line 22). To accomplish a task φ_i, a graph search is used, iterating from the robot's current position to determine the element $W(i,r)$ (lines 23 and 29).

The robot then follows the defined sequence of cells by connecting waypoints (the centroids of shared facets between successive cells in Seq_r), resulting in a piecewise linear trajectory suitable for an omnidirectional robot. Additionally, to ensure that each task φ_i is satisfied, the robot's trajectory is adjusted to include the centroid of every visited cell c_{nr}, ensuring full entry into the cell rather than reaching its facets (line 27).

Each robot individually follows its trajectory $Trajectory_r$, collectively achieving the global mission φ in a distributed manner. Algorithm 3 provides a centralized path-planning framework that serves as a foundation for trajectory-following routines while enabling decentralized execution of robot movements.

Remark 3.2 Individual robot trajectories may intersect. In practical scenarios, collision-avoidance mechanisms, such as priority-based waiting rules or resource allocation techniques [111], should be implemented.

The complexity of Algorithm 3 is influenced by the complexities of Algorithms 2 and 1, each of which is executed once. Additionally, there is a single invocation of MILP (3.2). The number of iterations is determined by the number of robots, $\mathscr{R}$, and the number of tasks, $|\rho|-1$ (as seen in lines 7, 19, and 22). In each iteration, the Dijkstra graph search represents the most computationally intensive part, making its relatively low complexity favorable for scalability with respect to the number of robots.

However, Algorithm 3 is not fully optimized for ensuring independence among robots. As a result, it may sometimes resort to the centralized solution from [107], even when the current formula contains independent tasks. This occurs because the algorithm considers only one decomposition from the *TaskSet* (rather than exploring all possibilities). Moreover, Algorithm 2 does not account for specific task sequences that may need to be assigned to the same robot.

Example 3.2.1 *Let us consider the following cost matrix* $W = \begin{bmatrix} 11 & 3 \\ 4 & 12 \end{bmatrix}$ *for the Example 3.0.1. By solving MILP* (3.2) *in step (ii), the allocation result assigns task* $\varphi_1 = b_1$ *to the first robot (red) and* $\varphi_2 = b_3 \vee b_4$ *to robot* r_2 *(blue). Consequently, robot* r_1 *must move to a cell within the region* y_1 *(set D for* φ_1 *containing all cells with observation* b_1*) and should traverse only through cells that do not intersect with region* y_2 *(according to set I, which includes all cells with observations other than* y_2*). Meanwhile, robot* r_2 *independently moves to a cell with an observation belonging to the set* $\{b_3, b_4\}$*, while avoiding* b_2 *(region* y_2*). As a result, the sequence* Seq_{r_1} *consists of 4 cells, and* Seq_{r_2} *contains 3 cells, as determined*

Algorithm 3: Overall solution

Input : Environment (sets $\mathscr{R}$ with their initial position of robots and $\mathscr{Y}$), LTL specification φ

Output : Individual trajectories in $Trajectory_r$

Obtain model for set $\mathscr{R}$ by executing Algorithm 1
Build Büchi automaton B for φ
Obtain *TaskSet* (the plausible decompositions of φ) by executing Algorithm 2
if $TaskSet = \emptyset$ **then**
    Obtain (synchronized) movement plans by using the from [107]
Select a decomposition $\bigcup_{i=1}^{|\rho|-1}\{\rho_B(\rho(i),\rho(i+1))\}$ from *TaskSet*,
for $i = 1,\ldots,|\rho|-1$ and $r = 1,\ldots,|\mathscr{R}|$ **do**
    $D_i = \{c \in \mathscr{C} \mid h(a) \subseteq \rho_B(\rho(i),\rho(i+1))$ and $mixed(c) = 0\}$
    $I_i = \{c \in \mathscr{C} \mid h(c) \subseteq \rho_B(\rho(i),\rho(i))\}$
    Build a graph based on the nodes $D_i \cup I_i$
    Perform graph search with Dijkstra from c_{S_0} to set D_i to return Solution
    if Solution $\neq \emptyset$ **then**
        $W(i,r) = min$ cost of Solution
    else
        $W(i,r) = N$
if $\exists i \in \{1,\ldots,|\rho|-1\}$ such that $W(i,:)\cdot 1 = N \cdot |\mathscr{R}|$ **then**
    Obtain (synchronized) movement plans by using the from [107]
Compute Z (robot-to-task(s) allocations) by MILP (3.2) solution
for $r = 1,\ldots,|\mathscr{R}|$ **do**
    $Seq_r = c_{S_0}$
    $Trajectory_r = x_{S_0}$
    for $i \in \{1,\ldots,|\rho|-1\}$ such that $Z(r,i) = 1$ *and*
     $W(i,r) \leq W(j,r), \forall j \in \{1,\ldots,|\rho|-1\}, j \neq i$ **do**
        Perform graph search with Dijkstra from c_{S_0} to set D_i to return Solution
        Let $c_{S_0}, c_{S_1}, ..., c_{nr}$ be the minimum cost path
        $Seq_r = Seq_r \cup \{c1, c2, ..., c_n\}$
        $Trajectory_r = Trajectory_r \cup Waypoint(p_{(k)r}, c_{(k+1)r})$, $k = 1,\ldots,n$
        $Trajectory_r = Trajectory_r \cup c_{nr}$
        $Z(r,i) = 0$ (task φ_i is solved)
        Update $c_{S_0} = c_{nr}$
Compute individual paths $Trajectory_r$, for $set\mathscr{R}$

by Algorithm 3. Figure 3.3 illustrates the trajectories by connecting the corresponding waypoints.. ■

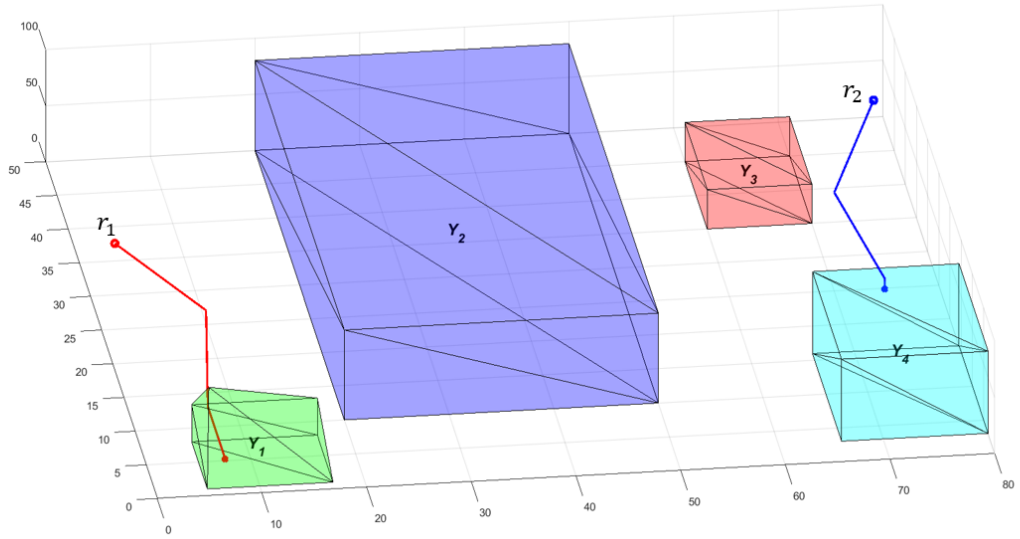


Fig. 3.3. Independent trajectories of the two agents, giving a solution to Example 3.1.1.

3.3 Numerical evaluation

Let us presents a numerical evaluation of the proposed solution for task decomposition and allocation in a multi-agent system. The algorithms are developed and executed in MATLAB on a laptop with an i7 - 8th gen CPU @ 2.20 GHz and 8GB RAM.

As noted at the beginning of the chapter, a 3D environment is considered, simulating motion planning for a team of UAVs (drones). Figure 3.4 illustrates the 3D workspace, which is divided into six regions of interest, $\mathscr{Y} = \{y_1, y_2, y_3, y_4, y_5, y_6\}$. Each region is a convex polyhedron with flat bases ($z = 0$), characterized by distinct shapes and heights. The regions are disjoint, except for y_2 and y_3, which intersect. The LTL specification, given in (3.3), requires that the robot team must eventually visit regions y_1, y_4, y_5, y_6, and the intersection of y_2 and y_3.

$$\varphi = \Diamond b_1 \wedge \Diamond (b_2 \wedge b_3) \wedge \Diamond b_4 \wedge \Diamond b_5 \wedge \Diamond b_6. \tag{3.3}$$

The two types of partitions described in Chapter 2.1 are referred to here as *Grid* and *OctTree*, respectively. The decomposition precision used is $\varepsilon = 16$. The model sizes and computational times for the two approaches are as follows: the *OctTree* method results in a model with 1849 nodes and 13081 transitions, computed in 6.7 seconds. In contrast, the

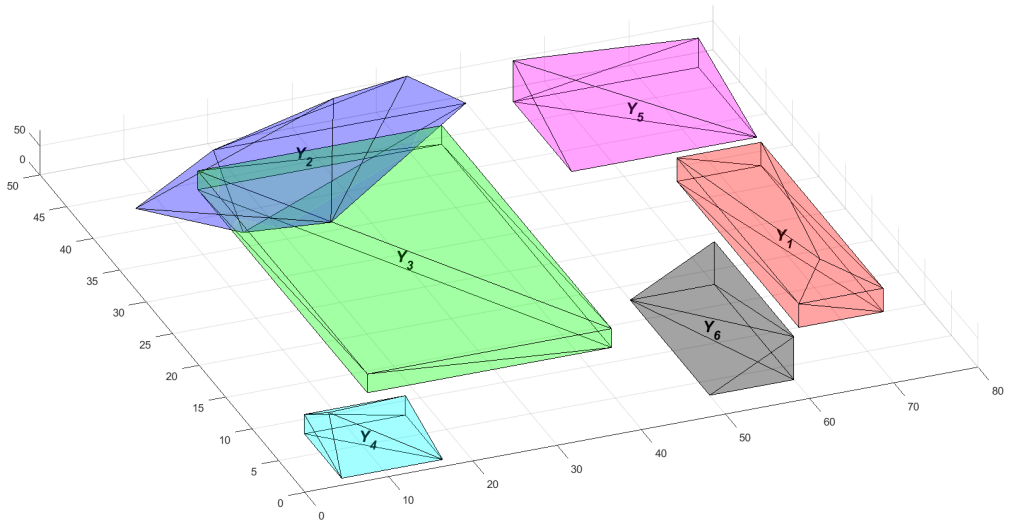


Fig. 3.4. Environment E with 6 regions of interest $\mathscr{Y}$.

Grid approach yields a model with 4096 nodes and 27136 transitions, requiring 19 seconds for computation.

Additionally, the simulations evaluate the proposed algorithm based on 100 experiments. In each experiment, the initial positions of the robots generate different sequences of cells and trajectory lengths, with small variations in computation times. For this reason, Figure 3.4 does not show the initial robot deployments. The results reported are the average values computed across the 100 experiments.

Table. 3.1. Average lengths of the trajectory for each agent, *OctTree* and *Grid* decompositions (Chapter2.1).

Average trajectory length	*OctTree*	*Grid*
r_1 $[lu]$	76.84	68.55
r_2 $[lu]$	86.02	69.56
r_3 $[lu]$	79.44	71.54

First, let us examine the execution times of various phases of the proposed method. The time required for trimming the Büchi automaton and decomposing the mission into independent tasks, as outlined in Algorithm 2, was 17.58 seconds. This execution time is independent of the environment partitioning method. The computation of the cost matrix W (averaged over 100 different initial robot deployments) took 6.84 seconds for the *Grid* method and 1.51 seconds for the *OctTree* method. As anticipated, the *Grid* approach took longer, as computing each cost involves a graph search over a subset of states from P_S.

Solving the MILP allocation took approximately 0.03 seconds, with this formulation being independent of the graph's size.

The second part of the comparison focuses on the performance of the resulting trajectories for the robotic team. Table 3.1 presents the average trajectory length for each agent. It is important to note that the trajectory length is influenced by the size of each traversed cell from Seq_r (Algorithm 3), with the optimization problem aiming to minimize the number of traversed cells, referred to as lu (length unit). The $OctTree$ partition typically includes larger cells, which might lead to longer trajectories compared to the *Grid* approach, but with the advantage of reduced computation times.

Table. 3.2. Average values for the maximum, respectively total costs.

Average cost	$OctTree$	*Grid*
Maximum cost	11.31	19.13
Total cost	27.52	47.10

Table 3.2 provides information on both the maximum cost and the total cost, as derived from the elements of W. The cost is defined as the number of cells traversed by the robot, which corresponds to the number of direction changes. The maximum cost in the first row represents the highest number of cells traversed by any of the three robots, with each robot completing all of its assigned tasks. The total cost refers to the cumulative number of places visited by all robots until the specification φ is completed. The fewer states in the $OctTree$ partition result in fewer direction changes during robot movement, even though the actual trajectory length might be longer.

For a clearer visualization of the trajectories, a video is available at [112], showcasing the movement of the robotic team in the 3D workspace. The simulations feature a team of three robots and five independent tasks, designed to highlight a scenario in which at least one robot is tasked with performing multiple tasks. More complex scenarios with a larger number of robots would produce trajectories that are more difficult to visualize.

A significant contribution of the LTL decomposition method is the implemented algorithm itself. To illustrate this, the cost of completing the mission from (3.3) is computed assuming only one drone is tasked with the entire mission, rather than distributing the tasks among the robots. For the $OctTree$ partition, a single drone would need to traverse an average of 34.6 cells, whereas for the *Grid* partition, it would need to traverse 68.74 cells. Comparing these values to those in Table 3.2, we can conclude that the decomposition method is beneficial in terms of both reducing the total number of traversed cells and enabling parallel execution of independent tasks, with the completion time reflected by the maximum cost in Table 3.2.

Chapter 4

Path planning with LTL specifications and path optimizing for multirobot systems

This chapter aims to provide an insightful description of a proposed framework based on the composition of the robotic team and the given mission. The main idea is to build a single model under the Petri net formalism, capturing both the motion of the robots. At the same time, their movement ensures the satisfaction of a given high-level mission. Particularly, this chapter establishes the foundation of a newly defined framework, denoted *Composed Petri net*. This representation models the high-level behavior of a homogeneous team fulfilling a global LTL specification requiring the reaching/avoiding of a set of regions of interest. A description of the theoretical formalism shall be given for each introduced model, accompanied by illustrative examples, for a better understanding of the proposed Petri net framework.

In the second part of this chapter, an efficient motion planning approach is included. The planning strategy is built upon a set of robotic trajectories and it ensures the motion execution while improving the quality of the paths. One contribution brought by this planning method is the parallel movement of the robots while maintaining the fulfillment of the mission, even in cases where the paths are rerouted. The planning is inspired by the Banker's algorithms, known for resource allocation and deadlock avoidance, associated here with the free space that the robots should share throughout their movement. Illustrative examples guide both the theoretically defined notions and numerical simulations validating the proposed solution. The evaluation of results considers comparison with a method from literature, for a scenario of a workspace where the robots should cross through a narrowed passage.

Based on the presented state of the art, this composition of a discrete event system associated with the motion of a robotic team with an automaton associated with high-level specification was never done before to solve such path planning problems. In addition, the

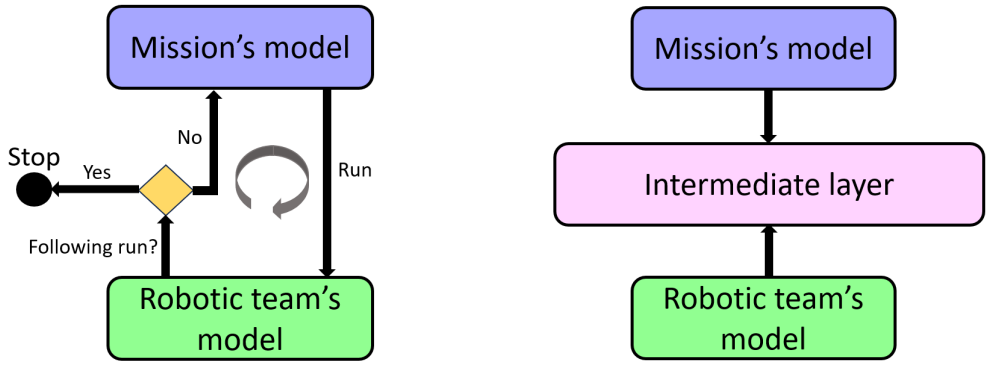
motion execution approach based on a known resource allocation algorithm represents a novelty in the robotic field. Thus, this chapter includes the following contributions:

- Novel framework combining the advantages of a Petri net modeling the motion of the robotic team, with the advantages of an automaton, respectively time automaton modeling a given specification under the LTL formalism.
- Introducing the concept of the intermediate layer to join two models, used in building scalable representations with respect to the number of agents through its fixed topology. The size of these models is represented by a sum of places rather than a product, e.g., as for automaton product [50].
- Defining an algorithm for parallel motion based on a predefined set of trajectories inspired by the Banker's Algorithm (BA), to navigate the robots in a constrained space efficiently, ensuring the global mission without deadlock.

4.1 Concept of an intermediate layer

The main idea of connecting two models into one model is driven by the joining the advantages of (i) the Petri net model of a team of agents, represented by a single model with a fixed topology when the number of identical agents is increasing or decreasing, with (ii) the automata model of a high-level mission capturing the space requirements, such as Linear Temporal Logic mission. The proposed framework builds a single Petri net model, an aspect that triggers the modeling of an automata into a Petri net.

Handling a single Petri net model for the path planning strategy of a multi-agent system is motivated also by a previous work [4], introduced in Chapter 2.4 under the abbreviation **FB**. Specifically, the planning method presents a motion planning strategy that utilizes the models of the robots, respectively of the mission, in a sequential manner. The method involves computing a set of k runs, represented as *prefix* and *suffix* components in the Büchi automaton of the given LTL specification, using the k-shortest path algorithm. Let us recollect that an LTL mission is satisfied only if there is an accepted run in the Büchi automaton, formed out of *prefix* and *suffix*. The solution to the planning problem consists of an algorithm that attempts to execute one of these runs following the Petri net model structure of the robots. There are situations in which the run in Büchi automata cannot be followed. One scenario is represented by the fact that the robots cannot generate the required observations, i.e., the disjoint regions of interest $y_i, y_j, i \neq j$ should be reached simultaneously (the atomic propositions b_i, b_j are evaluated as True), but there is only one robot in the environment. In another scenario, the observations triggered by the movement of the robots lead to other transitions being generated in the Büchi automata, without respecting the given run. Remember that this behavior might happen since the automata is nondeterministic. When these situations appear, the planning procedure is reiterated by selecting another run to be followed by the team.



(a) Workflow from previous work [4] (b) Representation of the composed framework

Fig. 4.1. Comparison of procedures motivating the composed framework

Figure 4.1 (a) illustrates a concise workflow of the work that was previously mentioned. One advantage that the current method brings is the use of the Petri net model for the robotic team, which is invariant with respect to the number of robots. Thus, the method offers an enhancement over the centralized approach based on transition system models that could increase exponentially in the state space as the number of robots grows. A significant limitation of this method is that the planning algorithm is incomplete; it cannot guarantee a solution even if one exists, due to its sequential approach. Moreover, the algorithm requires the computation of a set of k runs.

A solution for this hindrance is to have a more parallel approach compared with the previous work. The work [113] introduces a Petri Net (PN)-based approach within a "high-level" framework that ensures parallel execution of transitions in both the workspace Petri net and the Büchi Automaton corresponding to the Linear Temporal Logic (LTL) formula. This approach constructs a novel PN supervisor model, wherein a transition in the environmental PN is triggered only when a corresponding transition in the Büchi Automaton is satisfied.

The solution proposed in this thesis dwells on handling the Petri net model of the team and the Büchi automata of the LTL as one model. The composition is achieved through the commonalities of the models, which are represented by the set of atomic propositions addressing the spatial requirements of the robots. As visualized in Figure 4.1 (b), a set of places modeling the active and inactive observations. This composition facilitates the behavior of the robots such that their movements in the environment concerning the atomic propositions $\mathscr{B}$, e.g., regions of interest that should be reached, trigger a transition in the model of the mission. One benefit added by this solution is represented by collision-free trajectories, a characteristic that is not accounted for in the previous work [4].

One prerequisite that needs to be considered for the proposed framework based on composition, resides in the model of the robotic team, such that the places modeling the discrete space where the robotic team evolves are each labeled uniquely. This Petri net model

is denoted as *Quotient* model of the RMPN which is presented in [114] and is obtained by aggregating adjacent cells (represented by places) with identical observations.

For this, let us remember the observation function $h : P \to 2^{\mathscr{B}}$, assigning a label to each place based on the power-set $2^{\mathscr{B}}$, e.g., $h(p_i) = \emptyset, h(p_j) = b_1, p_i, p_j \in P$ representing the free space, respectively the region y_1. The unique values of the assigned labels ensure the activation of a new observation triggered by firing a transition. This aspect is essential in the synergy of the robotic team's model and the mission's model, considering their composition through the intermediate layer of places based on the active and inactive observations. Chapter4.2 presents in detail the explanation of this prerequisite, assisted by an algorithm and visual representation.

The proposed framework is under the Petri net formalism, denoted *Composed Petri net* and it concerns an LTL mission. Chapter4.2 describes the modeling workflow including also the algorithm to link a Petri net model to the Büchi automata, where the automata is translated into a Petri net model. Furthermore, the solution for motion planning is expressed by two Mixed Integer Linear Programming (MILP) optimization problems, showcasing an intuitive example in the results part. Fragments of this planning approach are published in [35].

4.2 Composed Petri net model

The first representation that is presented in this chapter is denoted *Composed Petri net*, capturing in one model both the space movements of the multi-agent team (defined previously by a reduced model of the full RMPN model of the environment) and the given global LTL mission. Firstly, the modeling workflow is introduced step-by-step supported by a visual representation of an easy-to-follow example. Afterward, the solution (in terms of robots' trajectories) is defined as a result of two proposed MILP optimization problems. The earliest MILP provides a solution in the reduced Quotient RMPN model, while the second MILP projects the solution into the full RMPN model. Lastly, a numerical example is presented considering the proposed framework, being also compared with the previous work [4].

Problem 1 *For a global LTL mission under the set of atomic propositions $\mathscr{B}$ and a robotic model represented as a Robot Motion Petri net (RMPN) (Definition 2.2.2), compute automatically trajectories of the robotic team such as the mission is ensured.*

For an easier understanding of this concept, let us provide a visualization of an environment that shall be used as an example throughout the explanations provided for the *Composed Petri net*. In addition, let us exemplify an LTL mission for this purpose, considering the defined workspace.

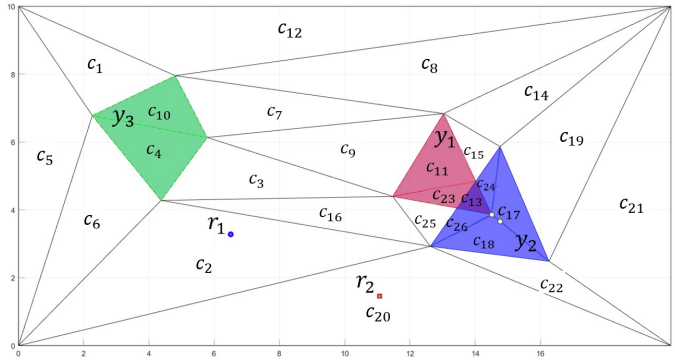


Fig. 4.2. Environment with three regions of interest and two robots

Example 4.2.1 *Let us consider the environment in Figure 4.2, containing three regions of interest ($\mathscr{Y} = \{y_1, y_2, y_3\}$) and two robots r_1, r_2.*

The workspace is divided into 26 cells, using the triangular decomposition previously defined in Chapter 2. The robots are initially placed in p_2 and p_{20}. Let us remember that a partitioned space can be represented by a RMPN model. This model consists of 26 places and 74 transitions, with the initial marking $m_0[p_2] = m_0[p_{20}] = 1$, the rest of the elements up to 26 being equal with zero. The observation function h highlights the cells that are associated with the regions of interest through the set of atomic propositions $\mathscr{B}$: $h(c_{11}) = h(c_{23}) = b_1$, $h(c_{17}) = h(c_{18}) = h(c_{24}) = h(c_{26}) = b_2$, $h(c_4) = h(c_{10}) = b_3$, $h(c_{13}) = \{b_1, b_2\}$, and $h(c_i) = \emptyset$ otherwise.

The given global LTL mission from equation (4.1) *convey to* eventually *visiting regions y_1, y_2 and y_3 while reaching y_1 and y_2 simultaneously.*

$$\varphi = \Diamond (b_1 \wedge b_2 \wedge b_3) \wedge \neg (b_1 \vee b_2) \mathscr{U} (b_1 \wedge b_2) \tag{4.1}$$

■

4.2.1 Modeling workflow

The core idea of the proposed solution can be broken down into two steps for clarity. In the first step, we derive a Petri net model that integrates an abstraction (Quotient) of the RMPN with the Büchi automaton, referred to as the *Composed Petri net* model. In the second step, a feasible run is identified within this new representation, producing a sequence of outputs that are accepted by the automaton. This run is then projected onto the original RMPN, and the robot trajectories are computed accordingly.

Various notations are used throughout the description of the proposed algorithm, which is divided into several steps. These symbols are essential when expressing the tuple $\mathscr{Q}$ for the RMPN and its components. Table 4.1 provides a summarized description to offer a clear

overview. In this context, we introduce a general notation for all components denoted as $< \cdot >$, with distinct symbols assigned to different topics.

Notation	Description
$< .^{M} >$	Denotes the variables used for Quotient RMPN (Sub-step 1.1)
$< .^{B} >$	Denotes the variables used for Büchi RMPN (Sub-step 1.2)
$< .^{C} >$	Denotes the variables used for *Composed Petri net* (Sub-step 1.3)

Table. 4.1. Notations for various PNs to be used

Figure 4.3 illustrates the steps and sub-steps of the proposed method for computing robot trajectories in a given environment to fulfill a global LTL mission. The first phase generates the *Complete Petri Net* model. Specifically, two Petri net models are utilized: (i) one for the environment (Sub-step 1.1), which uses a simplified abstraction of the entire space known as the Quotient RMPN, and (ii) one for the LTL specification (Sub-step 1.2), based on the Büchi automaton, which is represented as a Büchi Petri net. These models are then combined (Sub-step 1.3) into a compact representation that incorporates active observations through an intermediate layer, ensuring that the robots' movements adhere to the given specification.

The second phase focuses on producing the final solution. It involves two main actions: Sub-step 2.1, which generates the solution based on the *Composed Petri net* model, and Sub-step 2.2, which translates this solution into a sequence of robot trajectories. In Sub-step 2.2, the projection of the solution takes advantage of the fact that LTL is closed under stuttering [115], meaning that repeating the same input does not affect the truth value of the input string. Throughout the procedure, each step is supported by pseudo-code, MILP formulations, and a thorough depiction of the sound algorithm. The selected models in the approach combine the benefits of Petri net representation and the Büchi automaton, enabling a comprehensive solution for robot movements.

The Quotient RMPN net model, as described in [114], is used by aggregating adjacent cells that have identical observations. This model is then integrated with the Büchi automaton, which is converted into a Büchi Petri net to compute a feasible run. It is assumed that there are no active observations in the initial state, meaning that the robots start in free space, denoted as $\emptyset$.

Sub-Step 1.1. Quotient of the RMPN in Definition 2.2.2. Given an RMPN system $\mathscr{Q}$ as defined in Definition 2.2.2, the Quotient $\mathscr{Q}^{M}$ is derived by iteratively merging places p_i and p_j from P that satisfy the condition $p_j \in (p_i^{\bullet})^{\bullet}$ and $h(p_i) = h(p_j)$. This reduction method is detailed in Algorithm 4, and the reduced RMPN model $\mathscr{Q}^{M}$ is updated whenever a transition is triggered. As a result, when a transition occurs in $\mathscr{Q}^{M}$, a corresponding transition

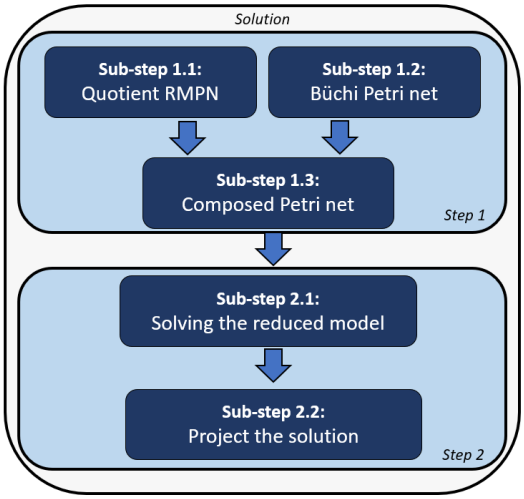


Fig. 4.3. Diagram of the global algorithm

in the Büchi automaton must also fire. Furthermore, since a transition in $\mathscr{Q}^M$ represents a set of trajectories in $\mathscr{Q}$, any sequence of transitions in $\mathscr{Q}^M$ can be mapped to a run in $\mathscr{Q}$.

The first four lines of Algorithm 4 initialize the elements of tuple $\mathscr{Q}^M$ based on those from tuple $\mathscr{Q}$. The main loop (lines 5 - 14) continues until no adjacent places with identical observations remain. In each iteration, for any pair of adjacent places $\langle p_i^M, p_j^M \rangle$ with the same observation, the transitions t_k^M and t_l^M representing the robot's movement from p_i^M to p_j^M and vice versa are removed (lines 6 - 8). Subsequently, the places p_i^M and p_j^M are merged into p_i^M, with updates made to the marking vector, the incidence matrices (lines 9 - 12), and the projection matrix Pr (computed in the lines 13 - 14).

Example 4.2.2 *Figure 4.4 portrays the discrete event system representations for the environment mentioned in Example 4.2.1: the top illustration is the full RMPN $\mathscr{Q}$ model based on the divided workspace with 26 cells, and the bottom picture is the reduced model Quotient RMPN $\mathscr{Q}^M$, as a result of aggregating states with the same observation (Algorithm 4).*

This updated PN model is also an RMPN, as defined in Definition 2.2.2, where each place represents a set of regions from the original RMPN system. It is important to note that Algorithm 4 also generates the projection matrix Pr, which in this case is a 5×26 matrix with all elements being zero, except for the following:

- $Pr[p_1^M, p_4] = Pr[p_1^M, p_{10}] = 1$, *indicating that place* p_1^M *combines* p_4 *and* p_{10}.
- $Pr[p_3^M, p_{11}] = Pr[p_3^M, p_{23}] = 1$.
- $Pr[p_5^M, p_{17}] = Pr[p_5^M, p_{18}] = Pr[p_5^M, p_{24}] = Pr[p_5^M, p_{26}] = 1$.
- $Pr[p_2^M, p_{13}] = 1$.
- $Pr[p_4^M, p_i] = 1$ *for all* $p_i \in P$ *with* $h(p_i) = \emptyset$.

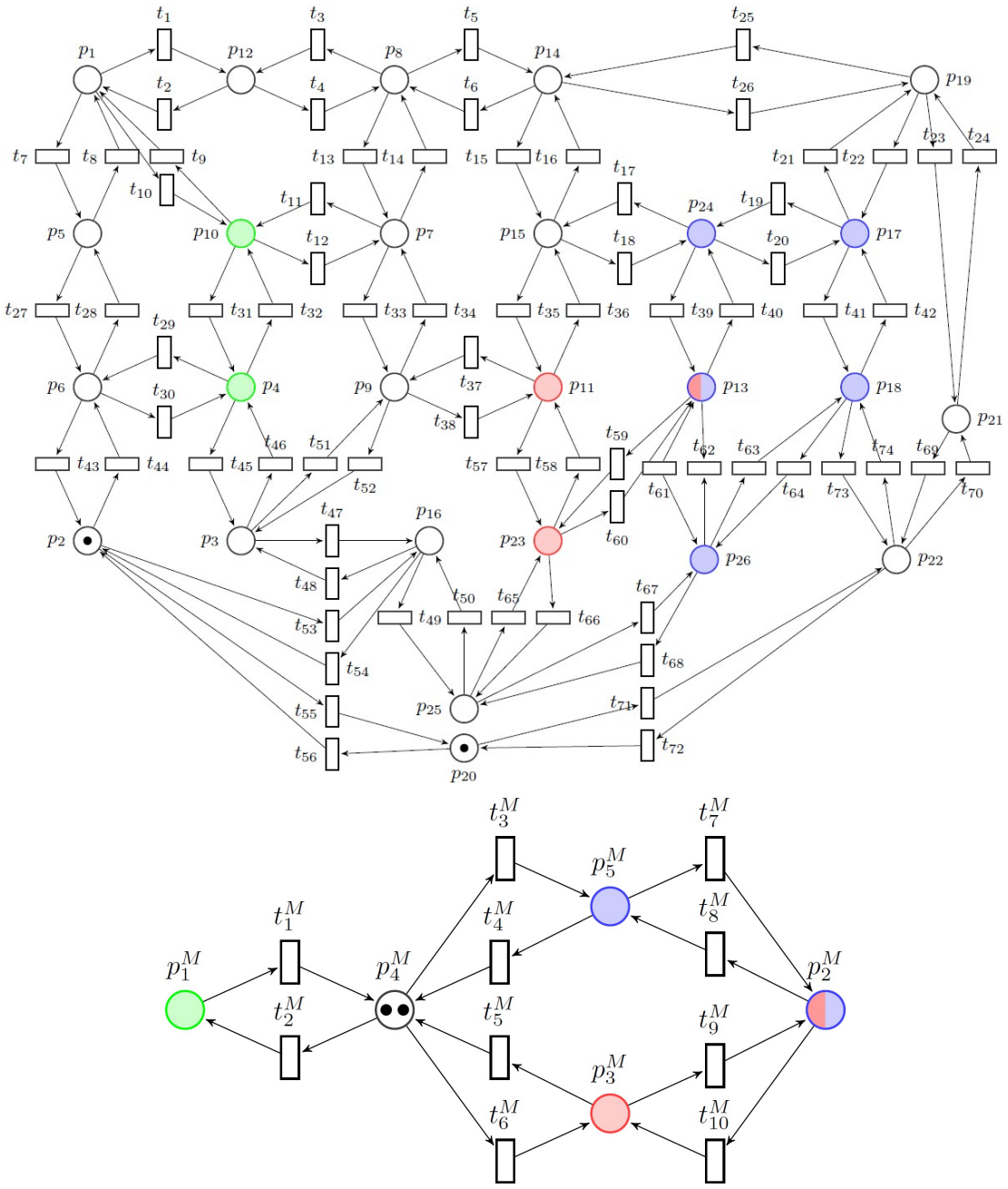


Fig. 4.4. Associated RMPN models of environment defined in Figure 4.2: (a) full RMPN $\mathscr{Q}$ - top, (b) Quotient RMPN $\mathscr{Q}^M$ - bottom

Algorithm 4: Quotient of RMPN system

Input : $\mathscr{Q} = \langle \langle P, T, Pre, Post \rangle, m_0, \mathscr{B}, h \rangle$
Output : $\mathscr{Q}^M = \langle \langle P^M, T^M, Pre^M, Post^M \rangle, m_0^M, \mathscr{B}, h \rangle$, Pr

1 $P^M = P$; $T^M = T$;
2 $Pre^M = Pre$; $Post^M = Post$;
3 $m_0^M = m_0$;
4 $Pr = I^{|P| \times |P|}$ /* Pr is the projection matrix */
5 **while** $\exists p_i^M, p_j^M \in P^M$ *such that* $p_j^M \in \left(p_i^{M\bullet}\right)^\bullet$ *and* $h(p_i^M) = h(p_j^M)$ **do**
6 Let $t_k^M = p_i^{M\bullet} \cap {}^\bullet p_j^M$ and $t_l^M = {}^\bullet p_i^M \cap p_j^{M\bullet}$;
7 Eliminate the columns associated with t_k^M and t_l^M from Pre^M and $Post^M$;
8 $T^M = T^M \setminus \{t_k^M, t_l^M\}$;
9 $m_0^M[p_i^M] = m_0^M[p_i^M] + m_0^M[p_j^M]$;
10 Remove the row p_j^M from Pre^M and $Post^M$;
11 Remove the element p_j^M from m_0^M;
12 $P^M = P^M \setminus \{p_j^M\}$;
13 $Pr[p_i^M, \cdot] = Pr[p_i^M, \cdot] + Pr[p_j^M, \cdot]$;
14 Remove the row p_j^M from Pr;

■

Sub-Step 1.2: Büchi Petri net. Starting from the Büchi automaton $B = \langle S, S_0, \Sigma_B, \rightarrow_B, F \rangle$ as defined in Definition 2.3.2, Algorithm 5 constructs the corresponding Büchi Petri net system $\mathscr{Q}^B$. For each state $s_i \in S$, a new place p_i^B is added to the Petri net (line 1).

The first loop (lines 3–8) processes each transition from s_i to s_j in the Büchi automaton. The second loop (lines 4–8) iterates over each conjunctive element α_k in $\pi(s_i, s_j)$ and adds a new transition t_{τ_k} to the Büchi Petri net, connecting p_i^B to p_j^B. Notably, all transitions corresponding to the conjunctive elements in $\pi(s_i, s_j)$ share the same input and output places in the Büchi Petri net.

In line 9, the marking vector is initialized and subsequently updated in line 10. Specifically, one token is added to the place p_0^B associated with the initial state $s \in S_0$ of the Büchi automaton. Since $|S_0| = 1$, a single token in the Büchi Petri net is sufficient to represent the current state of the automaton B.

Algorithm 5 also returns *virtual transitions* T^V, along with their relationship to the final states, which is encoded in Pre^V and $Post^V$. These transitions are essential for maintaining the Büchi Petri net in its final state, as required by the MILP in Equation (4.2) (discussed in the next section). One virtual transition is associated with each final state s_f of the Büchi Petri net (lines 11–15).

Algorithm 5: Büchi Petri net

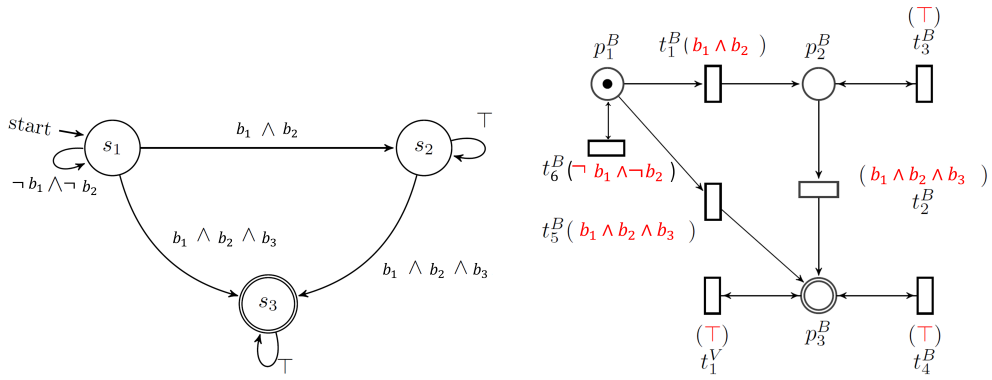
Input : $B = \langle S, S_0, \Sigma_B, \rightarrow_B, F \rangle$
Output : $\mathscr{Q}^B = \langle \langle P^B, T^B \cup T^V, [Pre^B\ Pre^V],\ [Post^B\ Post^V] \rangle, m_0^B, \mathscr{B}, h \rangle$

Let $P^B = \{p_1^B, p_2^B, \ldots, p_{|S|}^B\}$ be the set of $|S|$ places;
Let $T^B = \emptyset$ and $T^V = \emptyset$;
forall $(s_i, \tau, s_j) \in \rightarrow_B$ **do**
    forall *conjunctive element* α_k *of* $\pi(s_i, s_j)$ **do**
        $T^B = T^B \cup t_{\tau_k}$ /* add a new transition to T^B */
        Insert a new column into Pre^B and $Post^B$ associated with t_{τ_k};
        $Pre^B[p_i^B, t_{\tau_k}] = 1$;
        $Post^B[p_j^B, t_{\tau_k}] = 1$;
Let $m_0^B = 0^{|S| \times 1}$;
$m_0^B[p_0^B] = 1$;
forall $s_f \in F$ **do**
    $T^V = T^V \cup t_{s_f}$ /* add a new *virtual* transition to T^V */
    Insert a new column into Pre^V and to $Post^V$ associated with t_{s_f};
    $Pre^V[p_f^B, t_{s_f}] = 1$;
    $Post^V[p_f^B, t_{s_f}] = 1$;

Example 4.2.3 *Let us revisit the LTL mission from Example 4.2.1. The corresponding Büchi automaton for this task is shown in Figure 4.5(a), where the symbol* $\top$ *(True) represents any observation from* $2^{\mathscr{B}}$*. An accepted run satisfying the formula could be* $s_1, s_3, s_3, \ldots$*, with* s_1 *as the prefix and* s_3 *as the suffix. Note that the final state* s_3 *is visited infinitely often. However, as depicted in the environment of Figure 4.2, this run cannot be executed by the two robots. The robots are restricted to navigating through cells in the free space until one enters* p_{13}*, while the other simultaneously enters* p_4 *or* p_{10}*. Directly entering* p_{13} *is not feasible, as it requires first activating* b_1 *or* b_2*, which would violate* φ *from Equation* (4.1)*.*

A feasible run generated by the robots follows the sequence: $s_1, s_2, s_3, s_3, \ldots$*, with* s_1, s_2 *as the prefix and* s_3 *as the suffix. In this case, the robots must first synchronously enter* y_1 *and* y_2 *(producing* $\pi(s_1, s_2) = b_1 \wedge b_2$*). Subsequently, one robot must proceed to* p_{13} *(the intersection of* y_1 *and* y_2*). Finally, a robot must enter* y_3 *while crossing the free space to enable the transition to* s_3 *in the Büchi automaton. This solution is not unique, as the self-loop at* s_2 *allows for any valid input.*

The translation of the Büchi automaton into a Büchi Petri net is illustrated in Figure 4.5(b) using Algorithm 5. Boolean formulas for transitions are displayed in red. The places p_2^B *and* p_3^B*, corresponding to states* s_2 *and* s_3*, are connected by a single transition for the input* $\pi(s_2, s_3) = b_1 \wedge b_2 \wedge b_3$*. Additionally, the virtual transition* t_1^V *is connected to the final state* s_3 *via a bidirectional arc.* ■



(a) Büchi automaton (b) The associated Büchi Petri net

Fig. 4.5. Example of Büchi automaton and Büchi Petri net for the LTL formula in (4.1)

Sub-step 1.3. Composition of Quotient RMPN and Büchi Petri net systems. The entire approach for combining the robotic team model $\mathscr{Q}^M$ with the specification model $\mathscr{Q}^B$ into a unified framework, referred to as the *Composed Petri net* $\mathscr{Q}^C$, is detailed in Algorithm 6. This process requires an intermediate layer of places given by $2 \times |\mathscr{Y}|$, with half representing active observations (set P^O) and the other half representing inactive observations (set $P^{\neg O}$). Initially, places p_i^O contain zero tokens, while places $p_i^{\neg O}$ contain $|R|$ tokens, indicating the absence of active observations in the initial marking (line 3). The sum of tokens in p_i^O and $p_i^{\neg O}$ always equals $|R|$. Lines 1 to 2 detail the sets of places and transitions for the *Composed PN* $\mathscr{Q}^C$.

The initialization of matrices Pre^C and $Post^C$ is addressed in lines 4 to 5. For each observation b_i, lines 6 to 10 are executed. Input arcs are added to place p_i^O for each $p_k \in P^M$ where $b_i \in h(p_k)$. Furthermore, arcs from p_i^O to all output transitions of p_k are incorporated. This ensures that when a robot enters a region p_k with output y_i, a token is added to p_i^O, making b_i an active observation if $m[p_i^O] > 0$. On the other hand, $p_i^{\neg O}$ represents the complementary place of p_i^O, which is connected to the same transitions but with oppositely oriented arcs. If $m[p_i^{\neg O}] = |R|$, then b_i is inactive. Finally, the loop in lines 11 to 17 connects places p_i^O and $p_i^{\neg O}$ with transitions t_τ^B by reading arcs, following the assigned Boolean formula (conjunction), and considering weights of 1, respectively $|R|$.

Figure 4.6 illustrates a partial view of the full *Composed Petri Net* as built by Algorithm 6, including the initial marking. For clarity, this figure includes only the arcs associated with a specific region of interest (y_3). When a transition in $\mathscr{Q}^M$ fires and the observation b_3 changes, $\mathscr{Q}^M$ deposits a token into the respective active observation place. For instance, transition t_2^M is enabled when b_3 is not active (initially, the robots are in the free space), and if it fires, then a token is produced in both p_1^M and p_3^O. In $\mathscr{Q}^B$, transitions fire based on the assigned Boolean formula, triggered by the reading arcs from P^O and $P^{\neg O}$. For example, transitions t_5^B and t_2^B depend on the active observation b_3 of region y_3. The transitions in the Büchi PN in Figure 4.6 are color-coded according to the required active observations: red for

Algorithm 6: Composed Petri net system

Input : $P^O, P^{\neg O}$, $\mathscr{Q}^M = \langle\langle P^M, T^M, Pre^M, Post^M\rangle, m_0^M, \mathscr{B}, h\rangle$, $\mathscr{Q}^B = \langle\langle P^B, T^B \cup T^V, [Pre^B\ Pre^V], [Post^B\ Post^V]\rangle, m_0^B, \mathscr{B}, h\rangle$,

Output : $\mathscr{Q}^C = \langle\langle P^C, T^C, Pre^C, Post^C\rangle, m_0^C, \mathscr{B}, h\rangle$

Let $P^C = P^M \cup P^B \cup P^O \cup P^{\neg O}$;

Let $T^C = T^M \cup T^B \cup T^V$;

$m_0^C = \left[m_0^M,\ m_0^B, m_0^O\right]$;

Let $Pre^C = \begin{bmatrix} Pre^M & 0^{|P^M|\times|T^B|} & 0^{|P^M|\times|T^V|} \\ 0^{|P^B|\times|T^M|} & Pre^B & Pre^V \\ 0^{|P^O|\times|T^M|} & 0^{|P^O|\times|T^B|} & 0^{|P^O|\times|T^V|} \\ 0^{|P^{\neg O}|\times|T^M|} & 0^{|P^{\neg O}|\times|T^B|} & 0^{|P^B|\times|T^V|} \end{bmatrix}$;

Let $Post^C = \begin{bmatrix} Post^M & 0^{|P^M|\times|T^B|} & 0^{|P^M|\times|T^V|} \\ 0^{|P^B|\times|T^M|} & Post^B & Post^V \\ 0^{|P^O|\times|T^M|} & 0^{|P^O|\times|T^B|} & 0^{|P^O|\times|T^V|} \\ 0^{|P^{\neg O}|\times|T^M|} & 0^{|P^{\neg O}|\times|T^B|} & 0^{|P^O|\times|T^V|} \end{bmatrix}$;

forall $y_i \in \mathscr{Y}$ **do**

  Let $P' = \{p \in P^M | y_i \in h(p)\}$;

  forall $p_k \in P'$ **do**

    $Post^C[p_i^O, {}^\bullet p_k] = Pre^C[p_i^O, p_k{}^\bullet] = 1$;

    $Pre^C[p_i^{\neg O}, {}^\bullet p_k] = Post^C[p_i^{\neg O}, p_k{}^\bullet] = 1$;

forall $t_\tau^B \in T^B$ **do**

  Let π_i be the DNF formula assigned to t_τ^B;

  if $\pi_i \neq \top$ **then**

    forall *atomic propositions* y_i *appearing* ***not negated*** *in* π_i **do**

      $Pre^C[p_i^O, t_\tau^B] = Post^C[p_i^O, t_\tau^B] = 1$;

    forall *atomic propositions* y_i *appearing* ***negated*** *in* π_i **do**

      $Pre^C[p_i^{\neg O}, t_\tau^B] = Post^C[p_i^{\neg O}, t_\tau^B] = |R|$;

y_1, blue for y_2, and green for y_3. This rationale is consistently applied to the remaining active and inactive intersections, such as $b_1 \wedge b_2$.

Remark 4.1 The complexity of the Quotient PN is polynomial, specifically $\mathscr{O}(|P|^2)$, considering the partitioned environment. Algorithm 6 operates in polynomial time concerning the inputs $\rightarrow_B$ of the Büchi automaton B and the number of atomic propositions under set $\mathscr{B}$. Typically, Linear Temporal Logic (LTL) formulae involve a limited number of atomic propositions, and the exponential upper bound of $2^{|\mathscr{B}|}$ is rarely reached. Furthermore, Algorithm 6 is polynomial concerning the cardinalities of sets $\mathscr{B}$ and T^B.

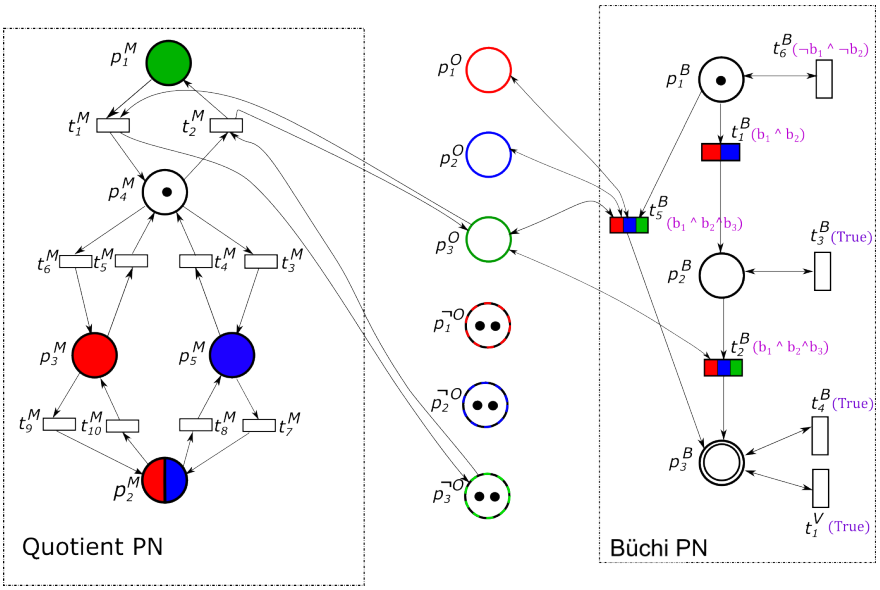


Fig. 4.6. Part of the Composed Petri net, based on active and inactive observations b_3 of y_3

4.2.2 Optimization-based solution

The planning algorithm for robot navigation requires achieving two steps: (i) computing a solution on the previously built model $\mathcal{Q}^C$ which considers a reduced representation of the robotic team, and (ii) projecting this solution into the full Petri net robotic model, returning the multi-robot system trajectories. This algorithm rules out the solutions that cannot be projected into the environment model based on the reduced solution (step (i)). This idea is similar to the one presented in [116], but it is applied to a PN-based rather than a graph-based approach.

Global Algorithm. Algorithm 7 calculates the movements of the robots while ensuring the fulfillment of the mission φ. The algorithm stops when it returns a set of trajectories for a place $p^B_{f_i} \in Set_f$, where Set_f represents the final states in B (line 15). The principal idea is to first search for a feasible run in $\mathcal{Q}^C$ (lines 3 - 12) using the MILP (4.2), which is executed separately for the *prefix* and *suffix*, each with a different initial marking. The *suffix* is computed by MILP (4.2) only when the last active observation of a final place $p^B_{f_i}$ is not included in its self-loop (i.e., the place p^O_j for observation b_j contains at least one token). If the last active observation is captured by the self-loop of the final place, then the suffix is defined by that final place. A feasible *Run* is obtained when both the *prefix* and *suffix* are non-empty (lines 11 - 15), after which a projection of this solution is explored using MILP (4.3).

If the *Run* cannot be projected into the original RMPN of the environment, the previous solutions returned by MILP (4.2) for both the *prefix* and *suffix* are saved, considering only the transitions in $\mathcal{Q}^M$. Line 17 appends these solutions $\sum_{i=1}^{k} \sigma_i^M$ of the prefix (denoted by

subscript $_p$) and suffix (denoted by subscript $_s$) to the sets of bad solutions $CE_{p|s}$. These sets are treated as counterexamples in MILP (4.2), forcing the computation of a different *Run*. If no overall solution is found for any p_{f_i}, the entire process is repeated with an increased number of steps k.

Algorithm 7: Global solution for robot's trajectories

Input : RMPN $\mathscr{Q}$, *Composed Petri net* $\mathscr{Q}^C$, the sets $\mathscr{B}$, $|R|$, P^O, Set_f, and the finite horizon k

Output : $Traj$ = Sequence of firing transitions

```
Let CE_{p|s} = ∅ and flag = False;
while (k ≤ U) OR (k > U AND flag = True) do
    forall p^B_{f_i} ∈ Set_f do
        Compute prefix for p^B_{f_i} with MILP (4.2);
        if prefix ≠ ∅ then
            Let P^O_f be last active observation for p^B_{f_i};
            if P^O_f ⊨ π(s_f, s_f) then
                Compute suffix with MILP (4.2), where m^C_0 = m^C_k ;
            else
                suffix = s_f;
        if (prefix ≠ ∅ AND suffix ≠ ∅) then
            Run = prefix suffix suffix ... ;
            Project Traj with MILP (4.3);
            if Traj ≠ ∅ then
                Return Traj;
        flag = True;
        CE_{p|s} = CE_{p|s} ∪ ∑_{i=1}^{k} σ^M_i;
    Increase k;
```

Sub-step 2.1. Solution on the reduced model. As previously mentioned, the solution in terms of robots' trajectories is returned by an algorithm considering two optimization problems. The first MILP (4.2), drives the Quotient PN to a state corresponding to a final state in Büchi (marking m^C_{2k}). The MILP is solved individually for *prefix* and *suffix*, each for k steps, with $k \geq 1$ being a design parameter. The odd steps are responsible for firing transitions in Quotient PN, while the even step triggers the firing of transitions in Büchi PN. The upper-bound of k is $U = \left(|P^M| - 1\right) \times \left(|P^B| - 1\right)$, as it may be necessary to move a token through all places P^M to produce a token in the next place Büchi PN.

Parameters:

- $|R|$ - number of robots;

- p_f^B - place modeling a final state in Büchi;
- C^C - token flow matrix of $\mathscr{Q}^C$;
- Pre^C - pre-incidence matrix of $\mathscr{Q}^C$;
- $CE_{p|s}$ - set of bad solutions for both prefix and suffix.

Variables:

- $m_i^C = \left[m_i^M\ m_i^B\ m_i^O\ m_i^{\neg O}\right]^T$ - marking column vector at step i of $\mathscr{Q}^C$ composed by the marking of Quotient PN (m_i^M), Büchi PN (m_i^B), active observation places (m_i^O) and inactive observation places ($m_i^{\neg O}$);
- $\sigma_i^C = \begin{bmatrix} \sigma_i^M \\ \sigma_i^B \\ \sigma_i^V \end{bmatrix}$ - firing vector at step i of the *Composed PN*, comprised by the firing vector of Quotient PN (σ_i^M), Büchi PN (σ_i^B) and of virtual transitions (σ_i^V);
- $z^1, z^2 \in \{0,1\}^{|T^M|}$ - binary vectors with $z^1[j] = 1$ if $\zeta - \sum_{i=1}^{k} \sigma_i^M \geq 1$, and $z^1[j] = 0$ otherwise, respectively $z^2[j] = 1$ if $\zeta - \sum_{i=1}^{k} \sigma_i^M \leq 1$, otherwise $z^2[j] = 0$, with $\zeta \in CE_{p|s}$.

Objective:

$$\min \sum_{i=1}^{2\cdot k} i \cdot \left(1^T \cdot \sigma_i^M + 1^T \cdot \sigma_i^B\right) \tag{4.2a}$$

Constraints:

$$m_i^C - m_{i-1}^C - C^C \cdot \sigma_i^C = 0, \qquad i=\overline{1,2\cdot k} \tag{4.2b}$$

$$\left.\begin{aligned} m_i^C - Pre^C \cdot \sigma_i^C &\geq 0, \\ 1^T \cdot \sigma_i^M &\leq |R|, \\ 1^T \cdot \sigma_i^B + 1^T \cdot \sigma_i^V &= 0, \end{aligned}\right\} \quad \begin{matrix} i=2\cdot j+1 \\ j=\overline{0,k-1} \end{matrix} \tag{4.2c}$$

$$\left.\begin{aligned} 1^T \cdot \sigma_i^M &= 0, \\ 1^T \cdot \sigma_i^B &= 1, \\ 1^T \cdot \sigma_i^V &= 0, \end{aligned}\right\} \quad i=2 \tag{4.2d}$$

$$\left.\begin{aligned} 1^T \cdot \sigma_i^M &= 0, \\ 1^T \cdot \sigma_i^B + 1^T \cdot \sigma_i^V &= 1, \end{aligned}\right\} \quad \begin{matrix} i=2\cdot j \\ j=\overline{2,k} \end{matrix} \tag{4.2e}$$

$$m_i^C[p_f^B] = 1, \qquad i=2\cdot k \tag{4.2f}$$

$$\left.\begin{aligned} \zeta - \sum_{i=1}^{k} \sigma_i^M &\leq N \cdot (1 - z^j), \\ -\zeta + \sum_{i=1}^{k} \sigma_i^M &\leq 1 + N \cdot z^j, \end{aligned}\right\} \quad \begin{matrix} \forall \zeta \in CE_{p|s} \\ i=\overline{1,2\cdot k}, j=\overline{1,2} \end{matrix} \tag{4.2g}$$

$$1^T \cdot (z^1 + z^2) \geq 1 \tag{4.2h}$$

Explanations for MILP (4.2) are as follows: (a) the objective function minimizes the number of transitions fired from sets T^M and T^B, favoring solutions that complete within the earliest possible steps k; (b) represents the state equation (2.1); (c) ensures robots advance at most one place in the Quotient PN when i is odd, leading to a change in observation ($h(p_i^M) \neq \emptyset$), and requires that in the subsequent step (i is even), a transition in the Büchi PN is fired; (d) enforces firing a transition from set T^B, ensuring progress from the initial state; (e) allows the firing of a single transition in the Büchi PN.; (f) guarantees that after k steps, the marking corresponds to the final state in the Büchi PN, denoted p_f^B; (g) and (h) ensure the current solution differs from any previously invalid solution ζ in the set $CE_{p|s}$, employing a big number (N) method [117].

Sub-step 2.2: Projecting the Solution. Let $M = \langle m_1^M, m_2^M, \ldots, m_{2k}^M \rangle$ denote the sequence of markings returned by MILP (4.2), with successive identical markings removed. Include m_0^M as the first element of M. Additionally, define $G = \langle g_1, g_2, \ldots, g_{2k} \rangle$, a sequence of $2k$ vectors such that $g_i \in \{0,1\}^{|P^M|}$, where $g_i[j] = 1$ if $m_i^M[j] = 0$, and $g_i[j] = 0$ otherwise. This vector ensures no additional observations are activated between consecutive steps.

MILP (4.3), used for projecting solutions, expands each marking in the Quotient RMPN (derived from MILP (4.2)) into a sequence of markings in the original RMPN while preserving the active observations to validate the LTL formula, even with finite repetitions. To prevent collisions, $|R|$ intermediate markings are introduced between successive markings, ensuring only one robot crosses each region at a time.

Parameters:

- M - sequence of markings computed by (4.2);
- Pr - the projection matrix between Q^B and RMPN models;
- C - the token flow matrix of RMPN model;
- $Pre, Post$ - the pre/post-incidence matrices of RMPN model;
- m_i^M - marking at step i of $\mathscr{Q}^M$.

Variables: $m_{i,j}$ - the marking at step i of RMPN model, based on the intermediate marking j; $\sigma_{i,j}$ - the firing vector at step i of RMPN, considering the intermediate marking j with $i = \overline{0,|M|}, j = \overline{1,|R|+1}$.

Objective:

$$\min 1^T \cdot \sum_{i,j} \sigma_{i,j} \tag{4.3a}$$

Constraints:

$$m_{i,j} - m_{i,j-1} - C \cdot \sigma_i = 0, \qquad i=\overline{0,|M|}, j=\overline{1,|R|+1} \tag{4.3b}$$

$$m_{i,0} - m_{i-1,|R|+1} = 0, \qquad i=\overline{1,|M|} \tag{4.3c}$$

$$Pr \cdot m_{i,j} - m_i^M = 0, \qquad i=\overline{0,|M|}, j=\overline{1,|R|} \tag{4.3d}$$

$$Post[g_i \cdot Pr, \cdot] \cdot \sigma_{i,j} = 0, \qquad i=\overline{0,|M|}, j=\overline{1,|R|+1} \tag{4.3e}$$

$$Post \cdot \sigma_{i,j} + m_{i,j-1} \leq 1, \qquad i=\overline{0,|M|}, j=\overline{1,|R|+1} \tag{4.3f}$$

$$m_{i,|R|+1} - Pre \cdot \sigma_{i,|R|+1} \geq 0, \qquad i=\overline{0,|M|} \tag{4.3g}$$

The constraints in MILP (4.3) are defined as follows: (b) represents the state equation of the PN (2.1); (c) ensures continuity of markings by requiring that the last intermediate marking matches the initial marking of the next step; (d) maintains the same observations across intermediate markings $m_{i,1}$ to $m_{i,|R|}$, as the cardinality of $|R|$ intermediate markings are introduced to prevent collisions between m_i^M and m_{i+1}^M.; (e) guarantees that the firing vectors corresponding to the intermediate markings from $m_{i,1}$ to $m_{i,|R|}$ do not activate additional observations; (f) enforces collision avoidance between consecutive markings in the original RMPN by limiting each region to be traversed by at most one agent between two intermediate markings; (g) requires that robot movements from substep $m_{i,|R|}$ to $m_{i,|R|+1}$ are synchronized, meaning all robots fire exactly one transition. This ensures that the resulting observation of $\mathscr{Q}^M$ changes consistently with the transitions fired in the Büchi PN.

Complexity. The use of MILp problems leads to an NP-hard algorithm. The total number of the unknown variables in both MILPs is based on the number of markings and transitions in both $\mathscr{Q}^C$ and $\mathscr{Q}$, and a set of binary variables of size $2 \cdot |T^M| \cdot |CE_{p|s}|$, bounded by U. The total number of constraints depends on the design parameter k for MILP (4.2), while for MILP (4.3) it depends on the previous solution.

Remark 4.2 The Algorithm 7 proves efficient in situations where a first returned solution by MILP (4.2) cannot be projected into the environment by MILP (4.3) (see Example 4.2.4. One downside is the fact that the proposed algorithm is not complete due to the possibility of spurious transitions as a result of MILP (4.2) and the collision avoidance restrictions of MILP (4.3), ensuring the correct solution represented by collision-free trajectories for the robotic team.

Example 4.2.4 *This example presents an illustrative scenario that demonstrates an unfeasible solution returned by MILP 4.2 within the planning method based on the defined Composed Petri Net model. In this case, the solution produced by MILP 4.2 cannot be projected into the environment by MILP 4.3 due to the partitioned environment. Thus, the proposed Algorithm 7 is designed to prevent these limitations, ensuring that feasible solutions are still provided.*

The projection step results from the collision avoidance constraint from MILP 4.3 thus preventing the robots from following the solution returned by MILP 4.2. The Algorithm 7 is

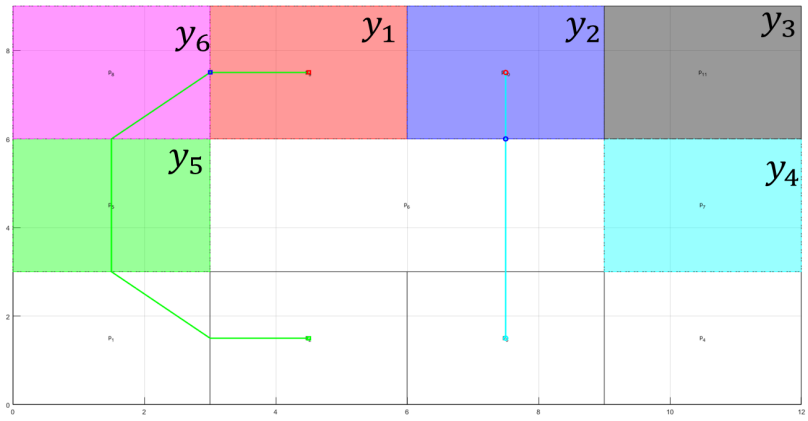


Fig. 4.7. Trajectories returned by the projection step (MILP 4.3) satisfying the LTL formula φ (based on the work from Chapter4.2

structured on the following guided idea: each time MILP 4.2 gives a solution that cannot be projected to the PN model of the robots (through MILP 4.3), that solution is added in a set of bad solutions, k (user-defined parameter representing the number of steps) is increased, and the procedure is iterated. Accordingly, MILP 4.2 has additional constraints that prevent any solution from the set of bad ones. The completeness is not achieved, yet, as the new MILP 4.2 could return firing vectors containing cycles, and eliminating such cycles or putting upper bounds on the number of cycles would require a huge number of additional constraints - an aspect that is not included since the obtained MILP would have been computationally intractable. The method is safe in terms of the possibility of returning feasible solutions. This method is implemented and integrated into RMTool under MATLAB [118].

Let us consider the example from Figure 4.7, considering a team of two robots and the LTL formula $\varphi = \neg(b_1 \vee b_2)\,\mathcal{U}\,(b_1 \wedge b_2)$*, imposing to reach the regions* y_1 *(color red) and* y_2 *(color blue) simultaneously. By design, the regions* y_3 *to* y_6 *surround the free space cell which is adjacent to both regions of interest* y_1 *and* y_2*. The robots could enter the required region* y_1, y_2 *either from the free space, or from the mentioned adjacent regions. The figure illustrates this scenario for which the first returned solution by MILP 4.2 cannot be projected by MILP 4.3. Both robots' trajectories are required to cross the free space while reaching the imposed regions of interest* y_1 *and* y_2*, results obtained by MILP 4.2 with respect to the shortest path in the Composed PN model. On account of the collision avoidance restriction in MILP 4.3, the latter solution could not be projected, as both robots were forced to pass through the same cell* c_6 *representing the free space.*

The proposed Algorithm 7 secures the robots' trajectories through another computed path, as a result of a new solution of MILP 4.2 which is different from the previous one demanding to reach y_1 *and* y_2 *directly from the free space. Therefore, the trajectories (highlighted with blue and green) express the current collision-free robotic path projected into the environment.* ■

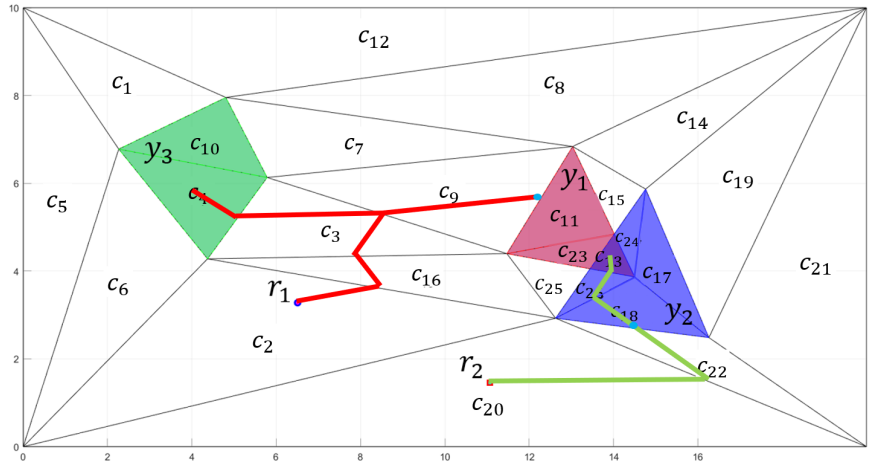


Fig. 4.8. Trajectories of the robots (r_1 - red, r_2 - green), ensuring the given LTL mission φ from equation (4.1)

4.2.3 Numerical example

The validation of the proposed framework *Composed Petri net* is shown here through numerical simulations, considering two examples: a simple one 4.2.5 based on the previously defined environment and LTL mission, and a more complex one 4.2.6 including a comparison between the proposed approach and a previous one [4]. The implementation of the model is integrated into RMTool - MATLAB [119], while the optimization solver selected for MILPs is CPLEX Optimizer [110]. The results were computed on a laptop with i7 - 8^{th} gen. CPU @ 2.20GHz and 8GB RAM.

Example 4.2.5 *In this example, the environment from Figure 4.2 is recalled with the LTL mission from* (4.1)*:* $\varphi = \Diamond (b_1 \wedge b_2 \wedge b_3) \wedge \neg (b_1 \vee b_2) \mathcal{U} (b_1 \wedge b_2)$*. The mission requires the robots to simultaneously reach all three regions of interest, ensuring that* y_1 *and* y_2 *are visited at the same time. Figure 4.8 illustrates the robot trajectories generated by Algorithm 7, with red and green indicating the paths of* r_1 *and* r_2*, respectively. Both robots first move toward regions* y_1 *and* y_2 *to satisfy the concurrent requirement. Subsequently,* r_1 *advances to enter the final region of interest,* y_3*, while* r_2 *moves into* $p_{13} = \{y_1, y_2\}$*.*

For this result, the Quotient PN model*, consisting of 5 places and 10 transitions, was computed in 0.02 seconds. The* Composed PN model*, comprising 14 places and 16 transitions (including one virtual transition), was also computed in 0.02 seconds. MILP* (4.2) *successfully reached the final state in the Büchi automaton within 0.05 seconds, using 180 unknown variables (*$k = 6$*). The returned solution included the prefix* $s_1 s_2$*, without requiring the computation of the suffix, as the final state* s_3 *had a self-loop with a* $\top$ *(True) condition. Projecting the solution took 0.05 seconds for 900 unknown variables, with a cost function value of 11, representing the total number of cells traversed by the robots.* ■

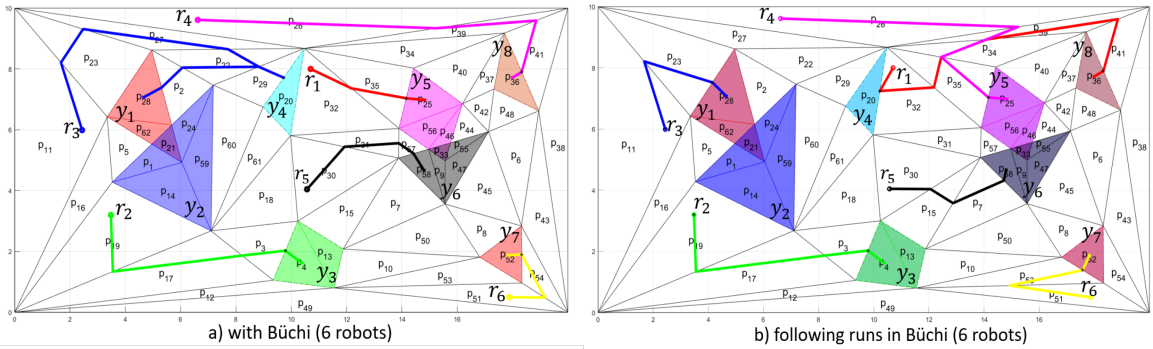


Fig. 4.9. Returned trajectories for Example 4.2.6 (red - r_1; green - r_2, blue - r_3, magenta - r_4, black - r_5, yellow - r_6)

Example 4.2.6 *For this example, let us consider the following LTL formula:*

$$\varphi = \square(\lozenge b_1 \wedge \lozenge b_3 \wedge \lozenge b_5 \wedge \lozenge b_6 \wedge \lozenge b_7 \wedge \lozenge b_8) \wedge \neg(b_5 \vee b_6)\,\mathscr{U}\,(b_5 \wedge b_6) \wedge \neg(b_4 \vee b_7)\,\mathscr{U}\,(b_4 \wedge b_7) \tag{4.4}$$

This specification requires visiting multiple regions of interest (ROIs) in an environment containing 8 ROIs. Additionally, the regions y_5 and y_6 must be visited simultaneously, as well as y_4 and y_7. Figure 4.9 illustrates the trajectories of 6 robots, with black stars indicating the synchronization points needed for the team to satisfy φ. ■

Table 4.2 contains a result analysis of the current approach of Alg. 7 in contrast with the work [4], previously described in Chapter2.4 under the notation **FB**. The comparison between methods accounts for two metrics: **(b) run time** to provide a solution and **(c) trajectory length** of the robotic team. The latter metric represents the value of the cost function (number of crossed cells of all robots) computed for $k = 10$ intermediate markings. One can refer to the present method as having a parallel approach, while the previous one can be considered as a sequential approach, with respect to the two models (robotic team and given specification). It should be noted that the collision avoidance strategy is ensured through the current planning strategy (restrictions (4f) of MILP (4.3)), contrary to the **FB** method out of which collision-free trajectories cannot be guaranteed.

Discussion. Based on the run simulations, this method of composing two models (RMPN of the environment and the Büchi automaton for the LTL specification) yields lower computational time and model size compared with **FB**. The current procedure returns a scalable PN model w.r.t. the number of robots, having a maximum number of places given by the following sum: $|P^M| + |P^B| + 2 \cdot |\mathscr{B}|$ (sum of places in the Quotient PN, the number of states in the Büchi automaton, and twice the number of individual observations). On the other hand, the model returned by the **FB** approach contains the number of places given by the RMPN $\mathscr{Q}$, while approaches based on transition systems **TS** [50] (mentioned in Chapter2.4)

Table. 4.2. Comparison between current approach and method **FB** [4] for Example 4.2.6

Number of robots	**(b) Run time to return a solution [sec]**		**(c) Trajectory length**	
	FB	*current approach: MILP* (4.2)	*FB*	*current approach*
4	9.56	0.75	37	39
5	2.22	0.26	23	28
6	0.77	0.11	20	21
10	1.2	0.78	13	13

are highly dependent on the size of the team. For example, the size of models in Example 4.2.6 are as follows: 37 places (*Composed Petri net*), 62 places (**FB**), $11^{|R|} \times 10$ (Quotient **TS**), as 11, respectively 10, represent the number of states in the reduced transition system, respectively in BA. The Quotient **TS** method refers to the product automata of all individual models of the robotic teams, where each state models a single observation, similar to the approach of Quotient PN.

Thus, the complexity is reduced as the number of places in the *Composed Petri net* model is smaller, compared with other approaches considered in the result analysis process.

4.3 Path rerouting considering parallel motion execution

Efficient task allocation and motion planning solutions maximize a team's productivity by assigning the right robot to the right task based on capabilities and workload [120, 121] while following the computed path. On the other hand, task reassignment is crucial in multi-robot systems operating in constrained environments, such as a narrow passage, as it allows the robots to dynamically adjust their tasks based on real-time conditions. Without reassignment, robots could collide, or delays might appear. By reallocating tasks, the multi-agent system ensures that robots efficiently reach their destination. Several papers tackle the problem of dynamic task allocation, either for UAVs [122], or in uncertain scenarios [123].

In narrow passages, space is limited, making it impossible for all robots to move simultaneously without coordination. Pre-assigned tasks may not account for changing spatial conditions, such as another robot blocking the way. If the robots fail to reassign their tasks, they may all attempt to cross the passage simultaneously, leading to a deadlock or inefficient movement patterns. The reallocation of tasks allows robots to prioritize certain tasks, such as letting the closest robot pass first, while others delay or reroute. This not only avoids collisions but also optimizes overall the robotic system performances by reducing wait times and unnecessary energy consumption.

The proposed solution enables parallel movement and collision-free paths for a team of identical robots in a known environment, while satisfying a global Boolean-based formula over a set of regions of interest. It builds upon the initial set of robotic paths provided by the approach in [2]. The parallel execution of the planning solution ensures deadlock avoidance, drawing inspiration from the Banker's algorithm, which is commonly used for resource allocation tasks. In this context, the resources are represented by the cells that model the free space shared by the robots during the mission.

The key advantages of the proposed solution are twofold: (i) it improves motion execution by reducing the number of waiting states of the robots along their paths, and (ii) it incorporates path rerouting, which is solved through a MILP problem, ensuring the mission is still fulfilled without generating new observations along the robot paths. To clarify, an *observation* (as defined in Chapter 2.1) refers to the *True* value of an atomic proposition from set $\mathscr{B}$ that is included in the global mission. Rerouting is triggered when the number of waiting robots exceeds a certain threshold or when stopped robots impede the movement of others. As a result, the method supports parallel robot movement and path rerouting when necessary. The algorithm is integrated into the open-source toolbox RMTool [119].

4.3.1 Driving factors

In the literature, the collision- and deadlock-free motion of robots is typically classified under Resource Allocation Systems (RAS). A widely adopted solution is based on the *Banker's Algorithm* (BA), which aims to prevent deadlocks. In essence, this algorithm simulates the maximum resource allocation (currently represented by the free space for the robots) for each process at every step, before deciding on the actual distribution of resources. This approach ensures that processes finish sequentially, though an iterative version can be applied where only one movement or operation is allowed per process step. It requires prior knowledge of the total number of resources in the system, and the system must be in a *safe state* at each step (meaning all processes can be completed within a finite time) [124].

While BA offers significant advantages for avoiding deadlocks, its application to mobile robot planning, where each robot's path is considered a separate process, is explored in only a few studies. Papers such as [125, 126] propose improvements to the Banker's Algorithm by incorporating graph representations, allowing robots to temporarily be in *unsafe states* under specific conditions, thus reducing unnecessary waiting times. An alternative approach is found in [127], which considers the dynamic release of resources during process execution. The benefits of using a Petri net system in conjunction with BA are discussed in [128], particularly in flexible manufacturing systems.

The current approach improves upon the more conservative method presented in [2], by introducing high-level path planning that allows for parallel movement of the robots. The method in [2] addresses the following problem: *Given a team of robots in an environment E and a global Boolean-based formula φ (as outlined in Chapter 2.3), find collision-free*

paths for the robots to satisfy the mission. This approach solves two MILP problems based on the team model RMPN (as defined in Definition 2.2.2): one for intermediate requirements and another for final requirements within formula φ. Additionally, both MILP problems assume a fixed number of intermediate markings (equal to the number of robots) to avoid collisions in the robot paths. The limitation of this approach lies in its sequential nature, where robots must reach goal cells one after the other when paths share *common resources* (defined below). In practice, this requires robots to navigate narrow passages, but the solution focuses solely on collision-free movement without considering parallel motion. This results in unnecessary waiting for some robots until others reach their destinations. Hence, a parallel motion strategy is needed, which will allow task reallocation, as further elaborated.

The primary goal of the current approach is to maximize the number of robots moving in a single global *step*, based on their predefined paths. The overall solution involves iteratively checking which robots can safely move to the next position. These robots must satisfy a global Boolean-based specification while taking advantage of the Petri net model in a partitioned environment. A key innovation in this approach is path rerouting without generating new observations along the paths, triggered by the number of robots waiting at a given step. Furthermore, rerouting is initiated when path blocking occurs due to parallel movement and robot waiting. This situation arises when some robots reach their final states but block the path for others. The rerouting is applied to the entire team, by solving a single MILP problem using the current and final markings of the Petri net system. This reduces the number of global *steps* needed to complete the mission.

The following assumptions are established, some of which are drawn from the previous work [2], while others introduce terminology relevant to the current planning strategy:

(a) The condition $m_0[p] \leq 1, \forall p \in P$ is satisfied, meaning that each robot is initially placed in a distinct partition cell (place of the RMPN).

(b) Each disjunction φ_i from the Boolean specification (Chapter2.3) can specify either regions to be visited or avoided.

(c) Each disjunction φ_i for intermediate or final requirements can be satisfied by a single robot deployment. In other words, there exists at least one RMPN marking that satisfies the requirements.

(d) Each disjunction for intermediate requirements may involve visiting multiple ROIs, or it can specify the avoidance of specific ROIs.

(e) An *obstacle place* refers to a place in the RMPN corresponding to a ROI that must be avoided, as required by the intermediate or final specifications in φ.

(f) A *step* is defined as a global time period during which at least one robot moves to an adjacent cell.

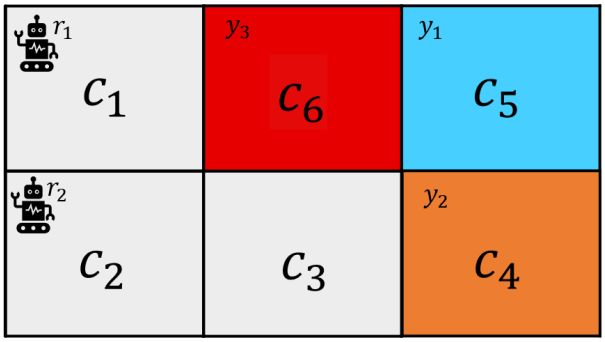


Fig. 4.10. Environment including 2 robots and 3 regions of interest

(g) A *resource* is defined as a cell in the path, while a *common resource* is a cell crossed by more than one robot along their paths. A similar interpretation of this concept is discussed in [129].

(h) A *robot collision* occurs when two robots occupy the same cell at the same time (step). The case of two robots swapping places is excluded, as will be detailed in the following section.

(i) A *process* represents the path of a robot, expressed as a sequence of cells, denoted as $Traj$. It is assumed that these (initial) paths are known and are provided by the solution in [2].

Therefore, the second part of this chapter addresses the motion planning problem as follows: *Given a set of paths, denoted Paths, for a homogeneous robotic team, and their evolution modeled as an RMPN 𝒬 system (Definition 2.2.2), compute collision-free parallel movements for the robots to satisfy a global Boolean-based specification (Chapter2.3).*

Example 4.3.1 *Consider the environment E depicted in Figure 4.10. The workspace is divided into 6 identical cells, represented by the set $\mathscr{C}$, containing 3 regions of interest: y_1, y_2, and y_3. Two robots, r_1 and r_2, are initially positioned in cells c_1 and c_2, respectively. For instance, cell c_6 corresponds to the third region of interest (y_3), so it is labeled as: $h(c_6) = b_3$. The labels for the other cells are as follows: $h(c_5) = b_1$ (for y_1), $h(c_4) = b_2$ (for y_2), and $h(c_1) = h(c_2) = h(c_3) = \emptyset$ (indicating free space), as these cells do not belong to any region of interest.*

Let us consider the following Boolean-based mission for the team of two robots:

$$\varphi = b_1 \wedge b_2 \wedge \neg B_3 \tag{4.5}$$

This Boolean specification (from Chapter 2.1) requires that region y_3 be avoided during the robots' movements (indicated in uppercase), while regions y_1 and y_2 must be visited at

the final positions (indicated in lowercase). The approach described in [2] generates two paths by assigning goal region y_1 to robot r_2 and goal region y_2 to robot r_1. The robots' paths are represented as $\langle (r_i, c_j), (r_{i+1}, c_k) \rangle$, where robot r_i is located in cell c_j and robot r_{i+1} is in cell c_k, both at the same step. In the initial step (step 0), the assigned resources for robots r_1 and r_2 are represented by cells c_1 and c_2, respectively.

$$
\begin{array}{llll}
Paths = \{ & & & \\
\quad \textit{step 0:} & \langle (r_1,c_1),(r_2,c_2) \rangle & \textit{step 1:} & \langle (r_1,c_1),(r_2,c_3) \rangle \\
\quad \textit{step 2:} & \langle (r_1,c_1),(r_2,c_4) \rangle & \textit{step 3:} & \langle (r_1,c_1),(r_2,c_5) \rangle \\
\quad \textit{step 4:} & \langle (r_1,c_2),(r_2,c_5) \rangle & \textit{step 5:} & \langle (r_1,c_3),(r_2,c_5) \rangle \\
\quad \textit{step 6:} & \langle (r_1,c_4),(r_2,c_5) \rangle & & \}
\end{array}
\tag{4.6}
$$

■

4.3.2 Algorithm for parallel motion

As observed, the robots move sequentially: r_1 begins its movement only after r_2 reaches its destination cell c_5. Due to this delay, the mission is completed in 6 steps (refer to (f) for the definition of *step*). To address the conservatism of this approach, we propose minimizing the number of steps required to complete the mission. The proposed solution enables collision-free parallel movement of the robots along their paths, while also rerouting them when necessary. This rerouting is achieved by solving a MILP based on the current (m_0) and final (m_f) markings in the RMPN model. By applying the strategy outlined in Algorithm 8 (described in detail below), the robots can reach their final destination in 4 steps. Their movements are represented as follows:

$$
\begin{array}{llll}
Paths' = \{ & & & \\
\quad \text{step 0:} & \langle (r_1,c_1),(r_2,c_2) \rangle & \text{step 1:} & \langle (r_1,c_1),(r_2,c_3) \rangle \\
\quad \text{step 2:} & \langle (r_1,c_2),(r_2,c_4) \rangle & \text{step 3:} & \langle (r_1,c_3),(r_2,c_5) \rangle \\
\quad \text{step 4:} & \langle (r_1,c_4),(r_2,c_5) \rangle & & \}
\end{array}
\tag{4.7}
$$

As previously stated, the solution reduces the steps required to fulfill the global mission by enabling parallel motion and reallocating tasks. Let $Traj = \{Traj_1, Traj_2, \ldots, Traj_r\}$ represent the set of robot trajectories, where $Traj_i$ corresponds to the path followed by robot $r_i \in \mathscr{R}$. The trajectory $Traj_i$ specifies the sequence of cells that robot r_i must traverse to reach its target cell. For instance, the trajectory of r_1 is expressed as $Traj_1 = [c_1, c_2, c_3, c_4]$. Although the method proposed in [2] effectively generates collision-free paths, it often results in multiple waiting instances in certain cells when paths involve shared resources, leading to increased steps. To address this, an algorithm is introduced to facilitate parallel execution along the trajectories in $Traj$ and reroute robots when excessive waiting occurs at specific steps.

Algorithm 8: Parallel motion of the robotic team

Input : $\mathscr{Q} = \langle \mathscr{N}, m_0, \mathscr{B}, h \rangle, m_f, Paths, N$
Output : $Paths'$ /* Planned robotic movement */

Build $Traj$ based on $Paths$;
Determine the order of resources $c_j \in P$ allocation to processes in $Traj$;
Remove obstacle place to update $\mathscr{Q}$;
$Paths' =$ step 0 in $Paths$;
while $(m_0 \neq m_f)$ **do**
    $RobotsToMove = \emptyset$;
    for $r_i \in R$ **do**
        c_j is the second place in $Traj_i$ /* the next robot to enter c_j is r_i */
        if *resource* c_j should be assigned next to r_i *AND* no robot in c_j **then**
            $RobotsToMove = RobotsToMove \cup \{r_i\}$;
        end
    end
    Update $Paths$ with the next step by moving $RobotsToMove$ to their next places while maintaining the other in their position;
    Remove first places of all $r_i \in RobotsToMove$ to update $Traj$;
    Update m_0;
    if $(|R \setminus RobotsToMove| \geq N)$ *OR* $(|RobotsToMove| == 0)$ **then**
        Solve MILP (4.8) to reroute paths considering m_0 and m_f;
        Build $Traj$ associated with the rerouted paths;
        Determine the order resources $c_j \in \mathscr{C}$ allocation to processes in $Traj$;
    end
end

The key idea is to modify the initial movements encoded in *Paths* by encouraging robots to move in parallel whenever feasible. This problem can be viewed as one of resource allocation, aiming to prevent deadlocks and collisions. Completing the processes corresponds to the multi-agent system reaching their designated goal cells. The strategy builds on the concept of a *safe state* from the Banker's Algorithm (BA), adapted to the current context. The proposed algorithm guarantees a solution under the previously stated assumptions. Initially, the order in which each process accesses resources is determined based on *Paths*, accounting for scenarios where certain resources are reused during a process (e.g., back-and-forth motions). This order ensures that all processes are eventually finished, as demonstrated in [2]. At each step, if a robot cannot move according to *Paths* but the next cell it should enter is free and the resource is assigned to the robot, the movement is shifted to the current step in $Paths'$. Furthermore, if N or more robots are waiting for resources at the same step, all trajectories are recomputed by solving the MILP (4.8).

Algorithm 8 outlines the complete process for robot motion, starting with *Paths* obtained using the approach from [2]. In addition to *Paths*, the user specifies a threshold N, which triggers rerouting whenever at least N robots are waiting during a step. Before enabling parallel motion, several preparatory actions are performed: compute robot trajectories in $Traj$ (line 1), determine the order in which robots cross cells (line 2), remove restricted areas along trajectories (line 3), and initialize the starting cells of robots at step 0 (line 4).

In line 6, the variable *RobotsToMove* is initialized as an empty set to store the robots moving in the current step. The loop between lines 7 and 12 checks whether each robot can move in the current step. A robot is allowed to move if the next cell is free and it is the robot's turn to occupy that cell. Once the set of moving robots is determined, lines 13–15 capture their parallel movements. The final section of the algorithm assesses the need for rerouting. If the threshold N is reached or no robot can move in the current step, the MILP (4.8) is invoked to recompute paths, updating the starting positions m_0 based on the robots' current locations (lines 17–19). This approach prevents deadlocks, which could arise if rerouting changes the order in which robots reach their goal cells and obstructs others' paths.

To enable parallel motion, path rerouting is applied to all robots. The MILP (4.8) ensures that trajectories are completed in sequential order, avoiding collisions and deadlocks. The implementation, as outlined, promotes parallel movement by solving the MILP. The optimization variables include the initial markings $m_{0,i}$, the final markings m_i, and the firing count vectors σ_i for each robot $r_i \in R$. These vectors are also used in the cost function to minimize the weighted sum of the r firing count vectors. Unlike previous MILP formulations for robot motion strategies—where σ encompasses the firing sequence for the entire team—this approach individually accounts for each robot's execution. Minimizing this cost function ensures that two robots cannot swap positions, thereby satisfying assumption (h).

The MILP (4.8) is subject to the following constraints:

- Constraints (i) correspond to the state equation (2.1) applied to a sequence of markings $m_i, i = 1, \ldots, r$. This sequence of markings is utilized to prevent collisions.
- Constraints (ii) enforce that no more than one robot occupies a given place at any time, taking into account the final positions of the preceding $(i-1)$ robots and the initial positions of the remaining $(|\mathscr{R}| - i)$ robots.
- Constraints (iii) and (iv) ensure the determination of the initial markings $m_{0,i}$ and final markings m_i for all robots $r_i \in \mathscr{R}$, consistent with the initial marking m_0 and the final marking m_f of the RMPN team model.
- Constraints (v) complement the earlier constraints by guaranteeing that $m_{0,i}$ represents the marking for a single robot.
- Constraints (vi) define the types of the unknown variables.

$$
\begin{aligned}
& \min 1^T \cdot \sum_{i=1}^{r} i \cdot \sigma_i \\
\text{s.t.} \quad & m_i = m_{0,i} + C \cdot \sigma_i, \ _{i=1,2...r} && (i) \\
& Post \cdot \sigma_i + \sum_{j=1}^{i-1} m_i + \sum_{j=i+1}^{r} m_{0,i} \leq 1, \ _{i=1,2...r} && (ii) \\
& \sum_{i=1}^{r} m_{0,i} = m_0 && (iii) \\
& \sum_{i=1}^{r} m_i = m_f && (iv) \\
& \mathbf{1}^T \cdot m_{0,i} = 1, \ _{i=1,2...r} && (v) \\
& m_{0,i} \in \mathbb{N}_{\geq 0}^{|P|}, m_i \in \mathbb{R}_{\geq 0}^{|P|}, \sigma_i \in \mathbb{N}_{\geq 0}^{|T|}, \ _{i=1,2...r} && (vi)
\end{aligned}
\tag{4.8}
$$

Remark 4.3 The proposed algorithm has the following limitations: the chosen threshold for the number of waiting robots, N, does not guarantee a monotonic relationship with the total number of global steps in the team path planning process (see Table 4.3); additionally, in certain cases, the solution may yield the same robot motion as the approach presented in [2].

4.3.3 Numerical results

The algorithm has been implemented and integrated into the open-source toolbox RMTool - MATLAB [119]. The simulations were performed on a computer equipped with an i7 8th-generation CPU @ 2.20GHz and 8GB of RAM. The rerouting MILP (4.8) was solved using CPLEX [110].

Example 4.3.2 *Consider Example 4.3.1 with the Boolean-based formula from* (4.5)*. The mission requires the robots to reach regions y_1 and y_2 while avoiding y_3. The solution provided by the current method, for $N = 1$, is shown in Equation* (4.7)*. Using this approach, the number of steps needed to complete the Boolean-based mission is reduced compared to the method from [2], with the total steps being reduced from 6 to 4.*

In the first step, only r_2 moves, as its next cell (c_3) is unoccupied, allowing it to advance. Meanwhile, r_1 cannot move because its next cell (c_2) is still occupied by r_2. For this small example, the runtime of Algorithm 8 is negligible, approximately 0 seconds. ■

Example 4.3.3 *To better evaluate the quality of the solution, consider a scenario where all robots must traverse a shared free passage to reach their destinations. Figure 4.11 depicts a grid-based environment consisting of 5 × 5 cells, with 8 ROIs and 4 robots initially positioned in cells c_1, c_6, c_{16}, and c_{21} (in the first column from the left). The Boolean-based specification is given by $\varphi = \neg B_1 \wedge \neg B_2 \wedge \neg B_3 \wedge \neg B_4 \wedge b_5 \wedge b_6 \wedge b_7 \wedge b_8$, which requires avoiding the first 4 regions (y_1, y_2, y_3, y_4) during the trajectories and visiting the last 4 regions (y_5, y_6, y_7, y_8) at the destination cells.*

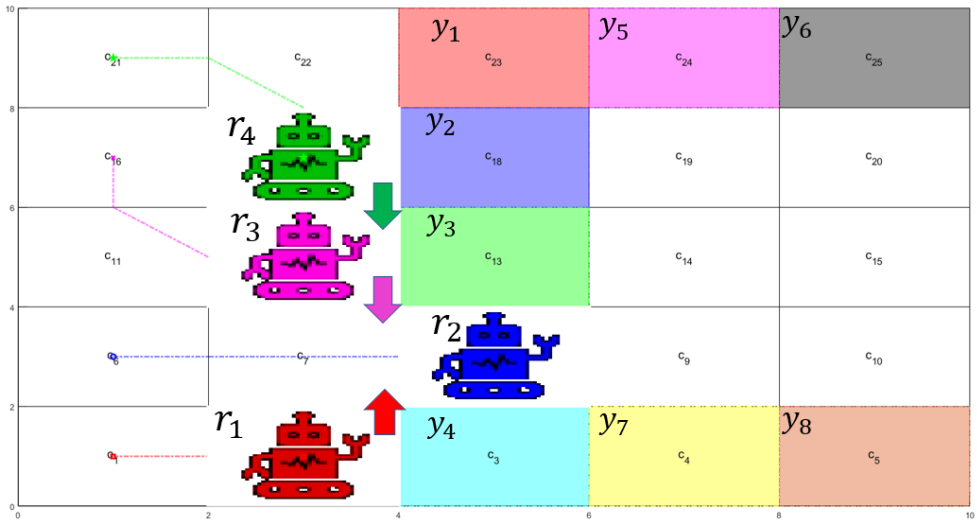


Fig. 4.11. Scenario for rerouting the robotic paths considering a grid environment with 4 robots and 8 ROIs

For this example, the computation time to generate robot trajectories using the approach in [2] is 0.1 seconds. The mission is completed in 24 steps with the previous method [2], compared to 11 steps achieved by the current approach. The parallel movement of the robots along their paths does not introduce noticeable additional computation time, particularly when rerouting is not required. The order in which the robots reach their final cells is detailed in Table 4.3.

For the specified threshold $N = 2$ (triggering rerouting when at least 2 robots are waiting in their current cells), the paths are rerouted twice using MILP (4.8)*, resulting in a total of 10 steps. Figure 4.11 illustrates the robot trajectories with dashed lines (r_1 - red, r_2 - blue, r_3 - pink, r_4 - green). The figure also shows the robots' current positions and their intended movements (indicated by colored arrows) immediately before the rerouting actions were initiated. The performance metrics for path rerouting, including the mean (μ) and standard deviation (std) of the running times, are summarized in Table 4.3.* ■

Table 4.3 presents quantitative results for scenarios involving shared resources and increased complexity, characterized by a larger number of cells and team size. The running time reported in the table corresponds to solving the respective optimization problem for each approach, as follows: the MILPs in [2] compute robot paths to satisfy the Boolean-based formula using a global model of the team. Meanwhile, the MILP in the current approach handles rerouting when necessary: *Case I* involves rerouting only when a final state is occupied but needs to be traversed by another robot, while *Case II* triggers rerouting when N robots are waiting. In both cases, the team's final cells are preserved, and collision-free paths are guaranteed.

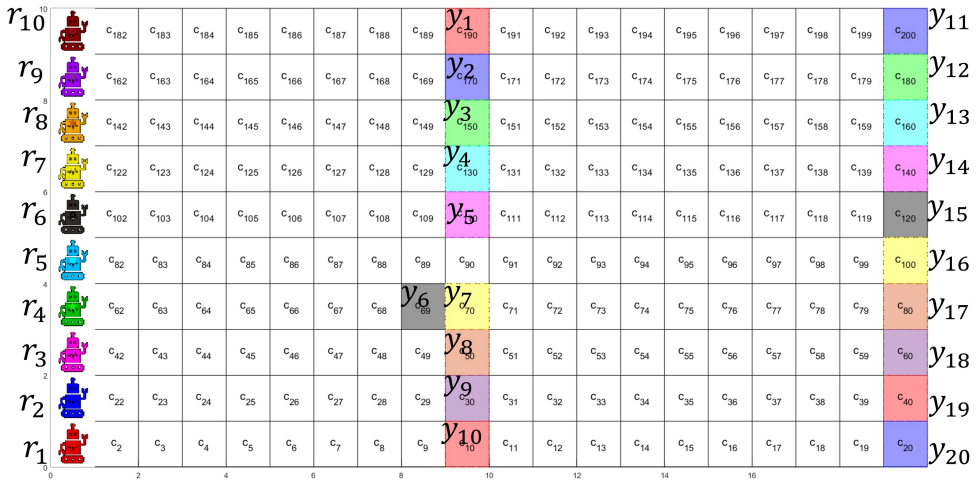
Table. 4.3. Numerical data comparison between the sequential approach [2] and the parallel (current) approach

Environment scenario	**Numerical performances**	**Planning approach [2]**	***Case I***	***Case I* or *Case II* (here $N = \lfloor \lvert\mathscr{R}\rvert/2 \rfloor$)**
Grid-based 5 × 5, with $\lvert\mathscr{R}\rvert = 4$ robots	Step count	24	11	10
	Count of rerouting actions	NA	0	2 times
	Execution time [sec]	0.1	0.1	0.1 + ($\mu = 7 \cdot 10^{-3}$, std = $1 \cdot 10^{-2}$)
	Order of robots to reach their final cell	r_3, r_1, r_4, r_2	r_2, r_1, r_3, r_4	r_2, r_1, r_3, r_4
	Assigned final regions for tupla (r_1, r_2, r_3, r_4)	(y_5, y_7, y_6, y_8)	(y_5, y_7, y_6, y_8)	(y_6, y_5, y_8, y_7)
Grid-based 10 × 20, with $\lvert\mathscr{R}\rvert = 10$ robots	Step count	155	39	43
	Count of rerouting actions	NA	0	6 times
	Execution time [sec]	3.2	3.2	3.2 + ($\mu = 1 \cdot 10^{-2}$, std = $1 \cdot 10^{-2}$)
Grid-based 20 × 20, with $\lvert\mathscr{R}\rvert = 20$ robots	Step count	352	78	7
	Count of rerouting actions	NA	2 times	14 times
	Execution time [sec]	12	12 0	12 + ($\mu = 9 \cdot 10^{-3}$, std = $3 \cdot 10^{-3}$)

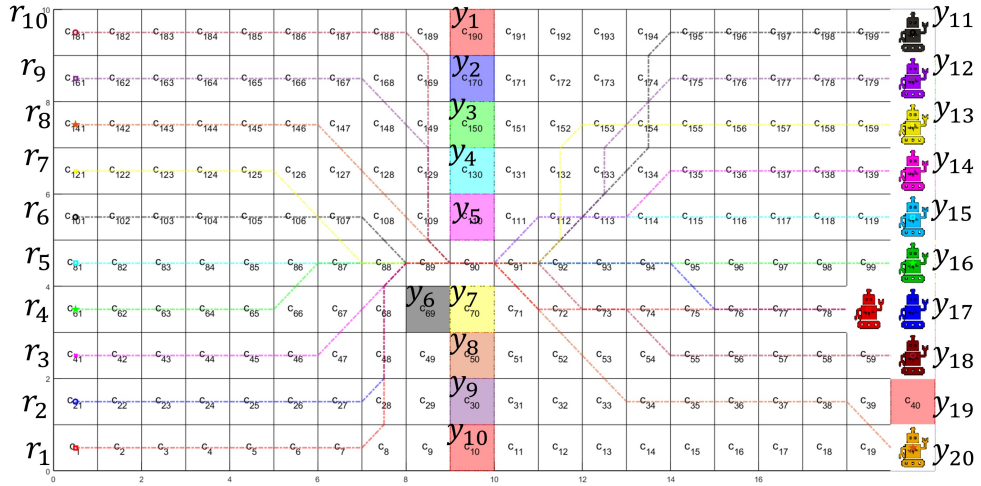
It is important to note that path rerouting is *Not Applicable* (NA) in [2], as this feature is a novel contribution of the current work. The first set of rows in the table reports results for Example 5, while the second set provides simulation results for the scenario discussed in [2], which involves a grid with 20 × 10 cells.

Figure 4.12 includes three stages from the entire movement of the robotic system, based on the proposed algorithm. The first figure shows the initial deployment of the robots. In this scenario, there are 20 regions of interest, out of which the first 10 regions should be avoided during the movement, while the rest of 10 should be visited as final destinations. As mentioned also at the beginning of this work, the rerouting procedure can be triggered by two cases. Figure 4.12(b) illustrates the second case. Particularly, it shows that one robot can be blocked by another robot, due to the previous task reallocation solution provided by MILP (4.8). The last image portrays the final destinations of the robots, as a result of their parallel movement. The first two scenarios are also captured in an animation that can be inspected here [130].

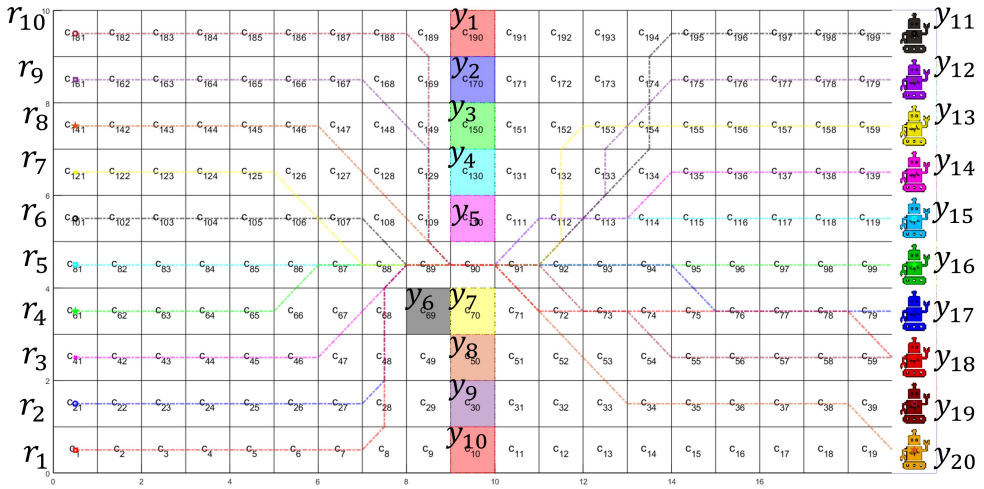
The third set of rows in Table 4.3 presents results for a grid-based environment with 20 × 20 cells and $\lvert\mathscr{R}\rvert = 20$. This scenario is not included in the animation recording due to reduced readability. In this case, the parallel movement for $N = \lvert\mathscr{R}\rvert$ involves two rerouting actions, triggered by certain robots being unable to move because other robots have already reached their final destinations and blocked some passages. When a trigger of $N = 10$ is applied for parallel movement, the robots reach their destinations in 77 steps. These results highlight the advantages of parallel robot motion with rerouting capabilities, particularly for large teams of mobile robots.



(a) The initial position of the robots



(b) The blocking scenario caused by the paths' rerouting procedure



(c) The final position of the robots as a result of task reallocation and the paths followed

Fig. 4.12. Example for a grid decomposition with 20×10 for 10 robots, 10 regions to avoid, and 10 regions to reach

Chapter 5

Path planning with MITL specification for multi-robot systems

This chapter extends the proposed Petri net framework previously introduced, by incorporating time constraints towards both the behavior of the robotic team, as well the mission expressed through an MITL specification. The framework is denoted *Composed Time Petri net*. The novelty of this model includes multiple contributions, such as (i) a planning strategy based on on-the-fly model-checking approaches that do not explore the entire state space of the motion problem, (ii) tailoring the model for returning trajectories for heterogeneous robotic systems, where each robot satisfy an individual MITL mission. Moreover, the second contribution presents a synchronization mechanism between the MITL missions, considering a *Composed Time Petri net* for each robot that are connected through a fixed Time Petri net topology. Illustrative examples accompany the proposed framework, evaluating the results through simulations. In addition, Chapter 7 presents an experimental setup that validates this model for a manufacturing application.

5.1 Modeling workflow

This section provides insights into the modeling strategy, with a focus on preserving the time expressiveness of the time automata of an MITL specification into a Time Petri net, as both models exhibit time constraints [100]. Compared with the *Composed Petri net*, where the solution is given by optimization problems, here a model-checking procedure is conducted to output the solution in terms of agents' trajectories. Parts of this chapter are included in [60, 30].

Specifically, this model captures not only the space requirements (as it was previously described in the previous chapter, where the robots ensure a global LTL mission), but also time requirements constraints which can be expressed through MITL specifications, e.g., "Reach region y_1 in 10 time units". The proposed model is denoted *Composed Time Petri*

net. The main idea of building this model is similar to the *Composed Petri net* approach considering the time Petri net representation defined in Chapter 2.2.2. The challenge here is to synchronize the local clocks in the robotic team with the time constraints specified under the MITL formula. The step-by-step workflow is introduced first, following a numerical evaluation assessing the proposed model in the second subsection, based on model-checking procedures.

Problem 2 *Consider that the motion of a robotic team is represented by a time Petri net model, as defined in Definition 2.2.4. Given a global MITL specification over the set $\mathscr{B}$, automatically generate the robots' trajectories to satisfy the specification while meeting both spatial and temporal constraints.*

The results of the *Composed Time Petri Net* model offer key contributions when compared to the *Composed Petri Net*: the incorporation of time constraints for the multi-agent system and the provision of a planning strategy for scenarios that require multiple agents to occupy the same region of interest. Another difference accounts for the path planning strategy, where the previous method considers mathematical programming-based methods, while the current model considers model-checking procedures. Lastly, this framework encodes in the set of atomic proposition actions that the robots should operate, actions that are linked to a region of interest that the robots should achieve. The latter characteristic will occur mainly in examples and simulation results, to highlight the applicability of the model in real-life scenarios.

The *Composed Time Petri net* framework offers a planning strategy for multi-agent systems operating under time constraints. It accommodates scenarios where either the entire robotic team must fulfill a global MITL mission or sub-groups of agents with similar capabilities must complete a set of MITL missions. Figure 5.1 illustrates the proposed solution in a step-by-step breakdown, where two time Petri net models are combined: the environment model (step 1) and the MITL specification model (step 2), forming a unified *Composed Time Petri net*. Consequently, a single *Composed Time Petri Net* model is created for each MITL specification. Although the composition of this model is similar to the structure of the *Composed Petri net* representation, the current model provides a distinct advantage by reducing the gap in utilizing time Petri net (TPN) models for motion planning. This is especially relevant in the context of using structural methods, such as mathematical programming, which remains an open problem in the literature.

Let us denote the *Composed Time Petri net* model as TPN^C. "C" is a superscript "C" added to every component of TPN^C, such as the following sets: of places P^T, of transitions T^T, and the input and output functions that define the arc weights between places and transitions $Pre^T, Post^T$. In addition, is added also to the marking m^T, and the function that maps a static interval to each transition I^T.

Step 1.1 In this step, the continuous environment is translated into a discrete representation, which simplifies the manipulation of the space E regarding the agents' motion.

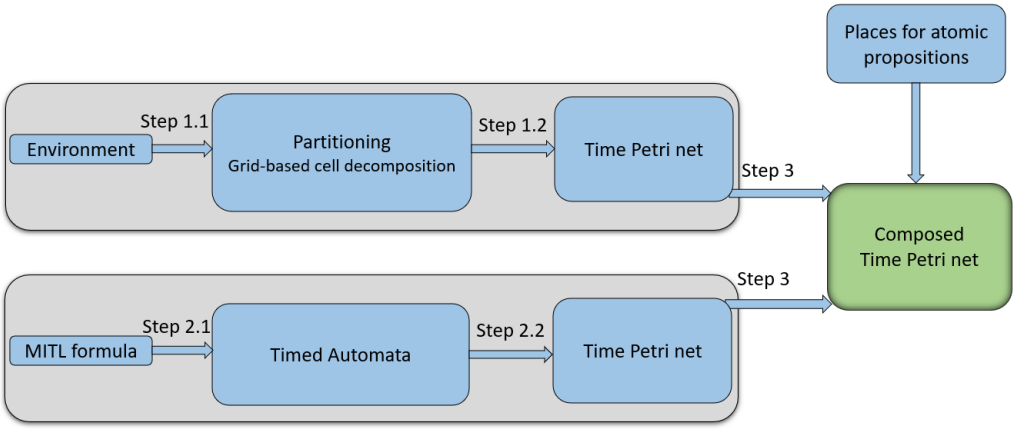


Fig. 5.1. General Framework of *Composed Time Petri net* model

The chosen mapping method is based on a cell decomposition technique, as discussed in Chapter2.1.

The robots' movement from one cell to an adjacent one is characterized by (i) the deployment of the robots in space (i.e., their position in the cell) and (ii) time constraints. These constraints are defined as $[\delta_{min}, \delta_{max}]$, where δ_{min} represents the minimum time to move, and δ_{max} is the maximum time required to reach the adjacent cell, considering the robot's minimum and maximum speeds. Note that this notation does not account for the waiting time in the current cell. If the agent has an undefined or unlimited time to wait in its current cell before moving to an adjacent cell, the upper bound is set to $\delta_{max} = \infty$. The movement from one cell to another is provided by a low-level control strategy, which can be user-defined. For instance, the movement can be described as traveling from the center of one cell to the center of an adjacent cell, passing through the middle of the shared facet.

Step 1.2 Once the environment E is partitioned, the robots' motion is modeled using a time Petri net, denoted as $\mathscr{TPN}^E$. The superscript "E" is applied to each component of $\mathscr{TPN}^E$. The desired TPN model is created by adding motion time constraints for movement between adjacent cells, as previously explained.

Following the building steps of the *Composed Petri net* model, the time Petri net model of the environment is partially Quotient, where adjacent places p_i^E and p_j^E, which share the same atomic proposition from the set $\mathscr{Y}$ (i.e., $h(p_i^E) = h(p_j^E)$), are merged, excluding the places that model free space. In this way, the time information related to the robots' movement is preserved. The output transitions from the merged places will have an upper bound of time set to ∞ to account for the robot's time in a region of interest, without being constrained by any specific time limit imposed by the MITL specification. In other words, the ∞ value indicates that the agent can stay in a region of interest for an unlimited amount of time. The lower bound is updated with the minimum time constraints derived from different

observations, reflecting the physical constraints of the robot when moving from one cell (place) to another.

Example 5.1.1 *Figure 5.2 illustrates an example for the first two sub-steps of the proposed workflow. On the left side of the figure, an environment E is partitioned into 4 cells, with two of these cells belonging to the same region of interest, y_1 (depicted in blue). On the right side, the corresponding $\mathscr{TPN}^E$ is shown, tailored as described earlier: both cells c_3 and c_4, with $h(p_3) = h(p_4) = b_1$ (representing region y_1), are modeled by a single place p_3^E. The token in place p_2^E indicates the presence of one agent in cell c_2.* ■

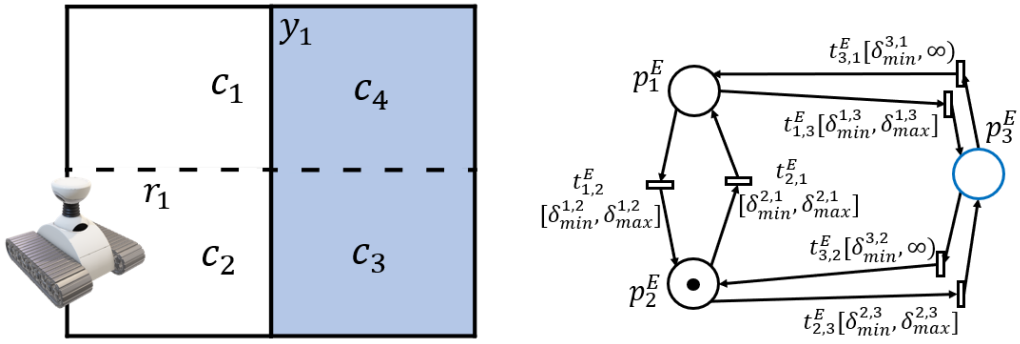


Fig. 5.2. Example of Time Petri net representation $\mathscr{TPN}^E$ (right) considering a partitioned workspace E (left)

The second phase of the proposed framework defines the mission of the multi-robotic system for an MITL specification.

Step 2.1 As outlined in Chapter2.1, according to Definition 2.3.5, any MITL formula φ in normal form (time interval $\leq \tau$ or $< \tau$, where $\tau \in \mathbb{Q}_+$) can be represented as a Timed B"uchi automaton (TBA) $\mathscr{A}$ (see Figure 5.3). Furthermore, any MITL specification can be converted to normal form by applying a set of four transformations, as demonstrated in [95].

Example 5.1.2 *Figure 5.3 shows an example of a TBA $\mathscr{A}$ that models the MITL specification $\varphi = \Diamond_{\leq \tau} b_1$, where $b_1 \in \mathscr{B}$ corresponds to the region of interest y_1, and $\tau \in \mathbb{Q}_+$ represents the clock constraint for the clock x. This mission requires reaching the region y_1 within τ time units. The clock reset is depicted in green. In this case, resetting the clock is optional, as it pertains to the temporal operator eventually. The clock reset becomes trivial in nested MITL specifications [102].*

The initial state q_0 is represented by the initial input arc, while the final state q_1 is indicated by the double edge, and q_2 denotes the sink state (error). If the MITL formula is not satisfied, the automaton transitions to the error state. The set of edges is as follows: $E = \{(q_0, \neg b_1, x \leq \tau, \emptyset, q_0), (q_0, b_1, x \leq \tau, x := 0, q_1), (q_0, \top, x \geq \tau, \emptyset, q_2), (q_1, \top, T, x := 0, q_1), (q_2, \top, T, x := 0, q_2)\}$, where $T \in \mathbb{Q}_+$ where $T \in \mathbb{Q}_+$ represents any time duration, and $\top$

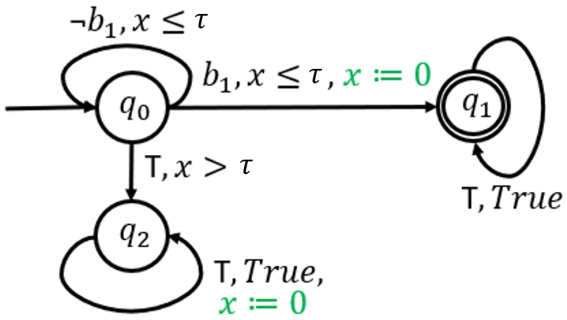


Fig. 5.3. Example of a Timed Büchi automaton accepting runs that satisfy the MITL specification $\varphi = \Diamond_{\leq \tau} b_1$

stands for the logical constant True, indicating that the edge can be triggered without any constraints on the value of the atomic propositions. ■

Step 2.2 The strategy here is to associate a time Petri net to a Timed Büchi Automata, necessary for the composition of the proposed model. Therefore, several papers examined their benefits in real-time systems, as well as their similarities and differences concerning timed bisimilarity and timed language acceptance.

Throughout the literature, two papers can be found relevant for this translation. By investigating their expressiveness, the first work [131] proposes a translation of a Timed Automata to a Bounded TPN with static priorities on transitions. Therefore, at one time instant, the transition with the highest priority will fire. For example, if transition t_1 has priority over t_2 in the time interval $[2, \inf)$ and both are enabled, then t_1 will fire at time 2. This approach encodes each atomic clock guard as Time Petri net models, then the nets are combined as a product. Afterward, several transitions are removed based on the combination of clocks in the TA. Furthermore, the invariants of TA can be also encoded as TPN models, while preserving the weak time bisimilarity property between the two representations. Since the TPN representation of the workspace does not consider priorities over transitions, another approach was exploited to translate a TBA to a TPN.

In [100], the authors considered a translation from a Timed Büchi automata with invariant clock constraints $x \leq \tau$ or $x < \tau$, with $\Phi(X) = \{x\}$ and $\tau \in \mathbb{Q}_+$, to a time Petri net model, taking into account the time bisimilarity property between the two representations. The main concept is to express the clock constraints and resets associated with each edge and each invariant as topologies within the time Petri net formalism. Each location of the automaton corresponds to a place, and each edge is modeled as a time transition, except for self-loop arcs defined as $e = (q, \top, T, \emptyset, q)$, where $q \in Q$. In other words, no transition is added for self-loops that do not have clock or logical constraints. Additionally, the transitions are connected to the TPN topologies assigned to clock constraints (represented by gray blocks) and clock resets (represented by green blocks), as seen in Figure 5.4. The full

algorithm is described in [100]. This TPN model corresponds to the TBA model shown in Figure 5.3, where the presence of a token in the final places $p_{fi}^{\varphi}, i = 1,2$ indicates the satisfaction of the time requirements.

The associated TPN structure for an MITL specification is denoted as $\mathscr{TPN}^{\varphi}$, with the superscript $\cdot^{\varphi}$ applied to all components of the model. The set of transitions $T^{\varphi} = T_c^{\varphi} \cup T_g^{\varphi}$, with $T_c^{\varphi} \cap T_g^{\varphi} = \emptyset$, consists of two sets: T_c^{φ}, which is used for the topologies of the clock constraints, and T_g^{φ}, which represents the transitions between the places corresponding to locations $q \in Q$ in the automaton $\mathscr{A}$. The labeling function for transitions is denoted as $\Lambda : T \to \mathscr{B}'_{\varepsilon}$. All transitions $t_{c_i} \in T_c^{\varphi}, i \geq 1$, are labeled as $\Lambda(t_{c_i}) = \varepsilon$ in order to simultaneously validate all clocks connected with transitions $t_{g_i} \in T_g^{\varphi}$, as per [100].

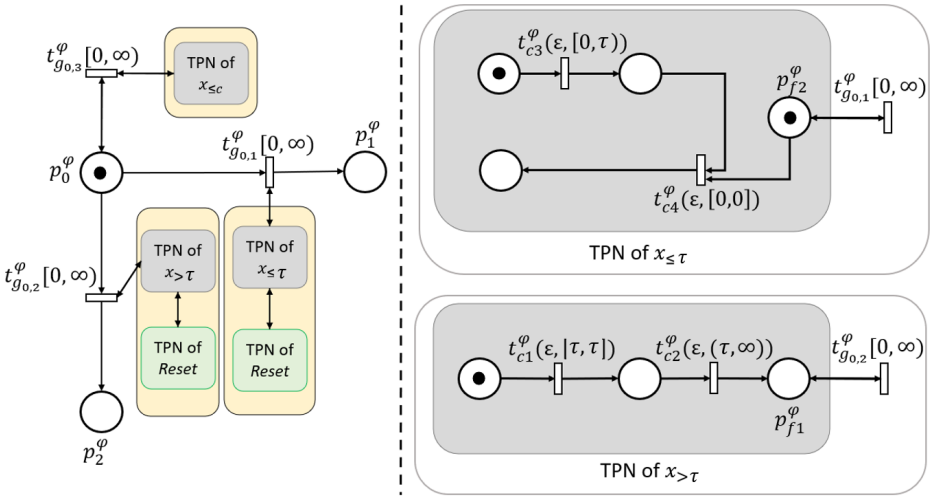


Fig. 5.4. An example of converting a TBA model to a TPN model (left) for the MITL formula $\varphi = \Diamond_{\leq\tau} b_1$, incorporating clock topologies (right)

Furthermore, we define the function $\Pi : T_g^{\varphi} \to 2^{\mathscr{B}}$ to store the symbols $\lambda \in \mathscr{B}$ assigned to the edges of $\mathscr{A}$. The values of this function are represented as a Disjunctive Normal Form (DNF) formula over the set $2^{\mathscr{B}}$, where $\mathscr{B}$ is the set of atomic propositions. In Figure 5.4, the transition $t_{g_{0,1}}^{\varphi}$ corresponds to the edge $e = (q_0, b_1, x \leq c, \emptyset, q_1)$, with $q_0, q_1 \in Q$, $b_1 \in \Sigma$, and $X = \{x\}$, hence $\Pi(t_{g_{0,1}}) = b_1$.

Step 3. To complete the construction of the proposed *Composed Time Petri net* model, denoted as $\mathscr{TPN}^{C}$, this step integrates the outputs from steps 1.2 and 2.2 along with additional inputs that model the atomic propositions from the set $\mathscr{B}$. As mentioned in Chapter 4.1, the sets P^{O} and $P^{\neg O}$ represent the true and false values of the atomic propositions, respectively. These places serve to provide a snapshot of the robots' movements with respect to the regions of interest they have reached. The sets P^{O} and $P^{\neg O}$ are included in $\mathscr{TPN}^{C}$ and function as an intermediary layer between the two TPN models, $\mathscr{TPN}^{E}$ and $\mathscr{TPN}^{\varphi}$.

The *Composed Time Petri net* model is constructed using the following inputs: the TPN model assigned to the robotic system, $\mathscr{TPN}^E$, with the labeling function for places h; the TPN model corresponding to the MITL formula, $\mathscr{TPN}^{\varphi}$, with the labeling function Π; the aforementioned sets P^O and $P^{\neg O}$; and the number of robots required to satisfy the MITL specification φ.

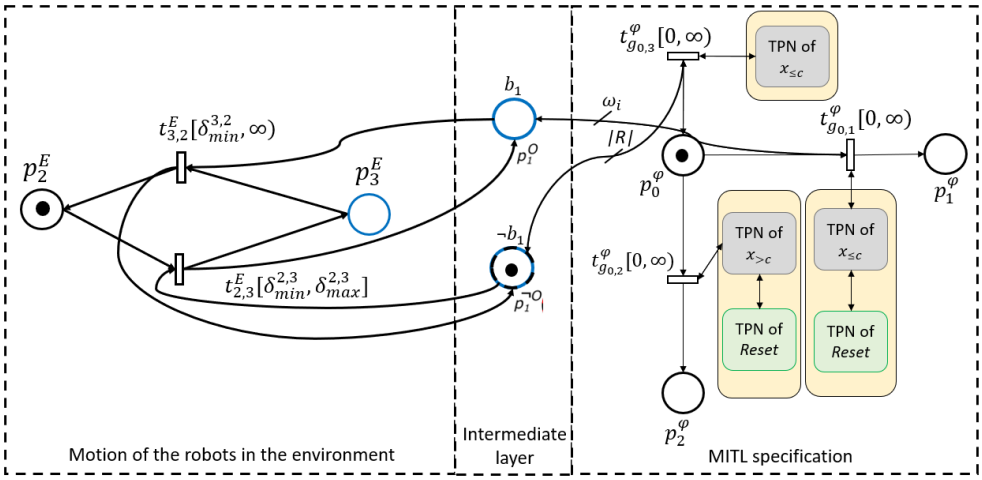


Fig. 5.5. Part of *Composed Time Petri net* model, for the atomic proposition b_1 included in the MITL specification $\varphi = \Diamond_{\leq \tau} b_1$

Example 5.1.3 *Figure 5.5 illustrates a portion of the Composed Time Petri net, which results from the overall workflow, providing a clearer view of the arcs that connect the two TPN models via the intermediate layer $P^O, P^{\neg O}$. The left side of the figure shows a partial time Petri net modeling the agents' movement within the environment, where $h(p^E_2) = \emptyset$ and $h(p^E_3) = b_1$, with b_1 being the atomic proposition associated with region $y_1 \in \mathscr{Y}$. On the right side, the time Petri net corresponding to the MITL formula $\varphi = \Diamond_{\leq \tau} b_1$ is displayed. The intermediate layer formed by sets $P^O, P^{\neg O}$ reflects the agents' positions in the environment: one agent is located in the free space, while no agent occupies the region of interest $y_1 \in \mathscr{Y}$. The linking arcs between the models $\mathscr{TPN}^E$, $\mathscr{TPN}^{\varphi}$, and the intermediate layer adhere to the previously outlined procedure. In essence, the atomic proposition b_1 is evaluated as True only if a minimum number of ω_i tokens are present in p^O_1, and it is evaluated as False when $|\mathscr{R}|$ tokens are found in $p^{\neg O}_1$.* ■

Remark 5.1 The current model is designed to provide solutions in scenarios where an MITL specification requires a minimum number of agents to fulfill the truth value of atomic propositions. This concept, referred to as "census" in the literature, involves having multiple agents execute the same task (in our case, the task is defined as reaching a region of interest). An example of the *census* concept is presented in [132], where STL (Signal Temporal Logic) specifications are used. Since the MITL formula is represented by an equivalent time Petri

net model, it helps in better understanding the *census* concept, which is represented by the imposed markings.

Example 5.1.4 *Consider a team of 5 robots and the MITL mission $\varphi = \Diamond_{\leq 5}(b_1 \wedge b_2)$, which express to visit both regions y_1 and y_2 simultaneously within 5-time units. Additionally, assume that the regions of interest should be reached by multiple robots, as mentioned in the previous remark. If one region, say y_1, must be reached by 3 robots, while region y_2 should be reached by 2 robots, the equivalent interpretation of the MITL specification in terms of the $\mathscr{TPN}^C$ model is: $\varphi = \Diamond_{\leq 5}((m^O[p_1^O] == 3) \wedge (m^O[p_2^O] == 2))$. This interpretation is used in the model-checking process to validate the model and compute the sequence of transitions needed to fulfill the MITL mission. Thus, a minimum marking is imposed on places p_1^O and p_2^O, which model the truth values of atomic propositions b_1 and b_2. These markings are sustained by the weight of the output transition from p_i^O and the incoming transition from $\mathscr{TPN}^\varphi$, representing the MITL mission. This weight is depicted in Figure 5.5 as ω_i on the arc between p_1^O and $t_{g_{0,1}}^\varphi$.* ■

The workflow differences between building *Composed Time Petri net* and *Composed Petri net* (Chapter 4.2) is mentioned below:

- Introducing time constraints into the $\mathscr{TPN}^E$ based on the movement of the multi-robotic system, as a result of steps 1.1 and 1.2;
- Modeling the TBA (Timed Büchi Automaton) of an MITL formula φ as a $\mathscr{TPN}^\varphi$ model, following the approach outlined in [100]. It is important to note that the translation from a TBA model to a TPN model, preserving the timed bisimilarity property, is possible only if specific conditions are met, as detailed in [100] and earlier in steps 2.1 and 2.2;
- Modifying the arc weights between places in the set P^O and transitions in T^φ, thereby enforcing the minimum number of agents required to achieve the true value for the atomic propositions in $\mathscr{B}$ (see Example 5.1.4).

5.2 Coordination mechanism for multiple Composed Time Petri net models

The main benefit of the Composed time Petri net model is that it combines the advantages of a specification MITL and the robot capabilities in its workspace. This section provides insights concerning a synchronization mechanism between multiple models, where individual MITL specifications are addressed to different robots. In this sense, the model can be associated with two scenarios: the MITL mission is given to a subgroup of agents, all of them required to fulfill the given mission under the *census* concept mentioned in the last remark, the MITL

formula is given individually for each agent, thus decoupling the tasks among the agents from the beginning. The last scene requires a coordination mechanism between the agents, as it will be detailed in the following. The versatility of this model allows for tailoring it according to the needs of the application:

- *(i) Atomic propositions* - Each MITL specification incorporates the time restrictions for a set of atomic propositions $\mathscr{B}$ defined by either a set of actions assigned to the robots, either by the regions of interest that the robots should reach under set $\mathscr{Y}$.
- *(ii) Robotic model* - The Time Petri net model of the robotic model embodies the individual or the collective space capabilities of reaching regions of interest for a single, respectively a subgroup of robots. Moreover, the model incorporates representations of the providing services in these regions of interest, based on a set of places and transitions.
- *(iii) Coordination* - Separate Composed Time Petri net models based on the given MITL missions are synchronized through a mechanism of adding a set of places and transitions, allowing for a global view in the entire robotic system.

The synchronizations of the robots assess the planned set of actions with their physical movements under time restrictions. Each action should be satisfied with respect to a time upper bound, thus resulting in a set of local clocks for the set of actions. The (a) high-level planner combines the restrictions of the mission with the kinematic capabilities of the robots, where the tasks cannot be executed immediately. Hence, the coordination of the clocks is ensured by adjusting the *Composed Time Petri net* models according to the needs of the applications, defined by a set of transitions fired instantaneously for the actions of each robot that should be synchronized. Each point enumerated previously is detailed below, emphasizing the adaptability of the high-level planning phase under Petri net formalism.

*(*i) The *Composed Time Petri net* model is defined considering the set of atomic propositions $\mathscr{B}$ to be associated with the set of regions of interest $\mathscr{Y}$ that the robots should reach under time requirements. Considering an application where a team of robots should collaborate, the multitude of individual tasks encapsulates the manipulating global mission that the robots should ensure. In Chapter 7.3 there is presented a manipulating application based on two cobots that receive individual MITL missions which combined, the multi-agent system should build a fixed structure. In short, one cobot receives a pick-and-place mission, while the second one receives a mission of putting the pieces together. Therefore, the set of atomic propositions is given by the set of actions that can be achieved by the robots to build a given structure, e.g. *PickTool*. For example, an MITL formula for a robot can be written as $\varphi = \Diamond_{\tau_i} PickTool$ expressing eventually picking up the tool in the time interval $\tau_i = [0, \tau_i]$. Notice that here, the MITL formula is based on atomic propositions expressed as actions, these actions being closely related to the region of interest that the robot should reach in order to ensure the specific task.

(ii) On the same note of achieving collaboration between robots based on their actions, a detailed explanation is needed for the robot's capabilities in the *Composed Time Petri net* model. The set of places P^E used in the robot's model is represented by the union of three subsets of places: (a) one set denotes places associated with the movement of the robot, based on the partitioned workspace (these places include also the association with the set $\mathscr{Y}$), (b) another set notes with the actions of the robots, captured also in the set of atomic propositions $\mathscr{B}$, and (c) the last set denotes the capacities of the robots related to the actions of the robots. In the case of (b), the places are connected to each Time Petri net model of the robots sharing the same action, e.g., two robots should pick up the same tool. The last set (c) handles the mutual exclusion [133] when one action is executed by one robot. For this, an even number of places are needed, due to the fact that one place expresses that the action can be made by any robot, and the complementary one portrays the fact that the action is conducted by one robot.

The two added sets from points (b) and (c) are necessary for the coordination of multiple agents, where their missions are given as individual MITL specifications. To reduce the use of new mathematical notations for these subsets, the explanations consider color codes, which are further detailed in the example.

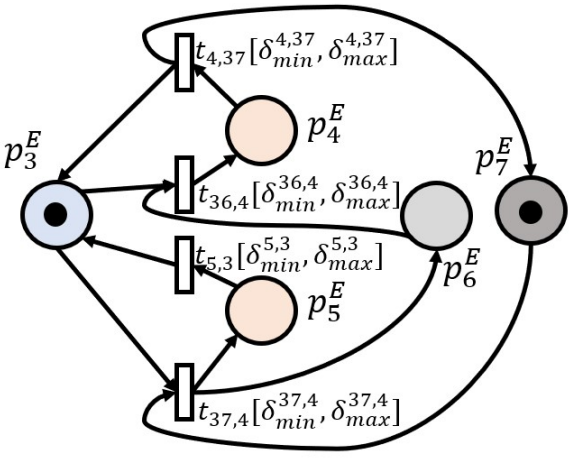


Fig. 5.6. Partial representation of the Time Petri net model of a robot

Example 5.2.1 *Figure 5.6 portrays a partial representation of the Time Petri net model of a robot, including all the places previously mentioned. Let us recall that the place* p^E_3 *models the region of interest* y_1*, with* b_1 *being its atomic proposition, illustrated with the blue color (from Figure 5.2). For simplicity, the rest of the places modeling the free space or other atomic propositions, are excluded from Figure 5.6. Let us consider that the robot should reach the region* y_1 *to pick a tool, e.g., a welding gun used to glue pieces together. Thus, places* p^E_5*, respectively* p^E_4 *(orange color), model the actions of PickTool, respectively*

PlaceTool. The presence of the token in p_3^E represents the robot being in region y_1. When the token is in places p_5^E or p_5^E, the robot picks up the tool or places it back after using it.

The gray places allow us to design mutual exclusion constraints used for this application in reasoning with the actions of a robot: place p_7^E has the capacity equal to one, imposing that a single robot can hold the tool at any time. The place p_6^E is the complementary place illustrating that the robot picked up the tool, a fact that ensures placing the tool as the continuity of the pick operation. Notice that these places maintain the logical course of the actions. Specifically, a tool cannot be placed if it hasn't been picked up previously. The time allocated to the transitions seize the complete action, e.g., picking up a tool from place p_5^E requires at least $\delta_{min}^{37,4}$ and maximum $\delta_{max}^{37,4}$ time units, the limits encapsulating the motion of the robot for the action PickTool based on its velocity. ■

The time restriction for a robot to fulfill a mission $\varphi = \Diamond_{\tau_1} PickTool$ depends on the robot's representation. This specification requires the robot to pick up a tool in less than τ_1 time restriction. If the previous example is considered, with the difference that the robot is in a region connected with y_3 (thus, no initial marking in place p_3^E), then the desired marking p_5^E is reached only if all the fired transitions up to $t_{37,4}$ following the imposed time τ_1. For example, considering the MITL $\varphi = \Diamond_{\tau_1} PickTool \wedge \Diamond_{\tau_2} PlaceTool$, the clock for τ_2 starts after the action *PickTool* is *True* in the time interval τ_1. Hence, the atomic propositions *PlaceTool* and *PickTool* are not commutative following the stated assumption previously described.

(iii) The *Composed Time Petri net* expresses the time requirements under the MITL specification that are embodied in the same model with the robot's representation. For an easier explanation of the synchronization mechanism between MITL specifications, the following example considers a scenario where two MITL missions are given to two individual robots. For each robot, a *Composed Time Petri net* is built, integrating both the robot's capabilities with the constraints from the MITL specification. The synchronization of the robots is achieved by adding a set of waiting places that triggers temporal transitions with time $[0,0]$ (instantaneous firing) between the relevant clocks associated with the time restrictions.

Example 5.2.2 *Specifically, let us consider the following MITL formulae $\varphi_1 = \Diamond_{\tau_1} b_1$ and $\varphi_2 = \Diamond_{\tau_2} b_2$ describing that r_1 eventually ensure reaching region y_1 in time τ_1 while the second robot r_2 eventually reaches region y_2 under time τ_2.*

Figure 5.7 illustrates the synchronization mechanism. Both Composed Time Petri net models are highlighted through this figure, maintaining the mathematical notations defined throughout the thesis. Let us assume that $\tau_1 = \tau_2$. For example, both robots should reach the regions y_1 and y_2 in 5 time units. Both MITL missions can be ensured if one robot reaches the region of interest in 2 seconds, and the second one in 3 seconds. The idea of this mechanism is to ensure that both robots synchronize. Thus, since the robots can ensure their missions

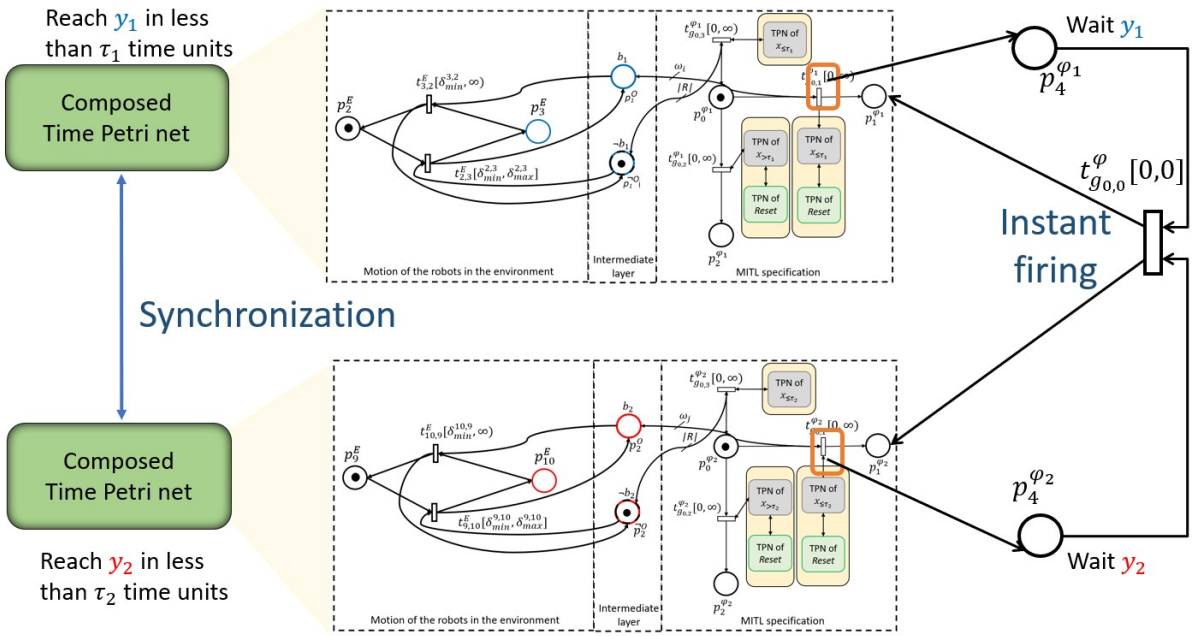


Fig. 5.7. Synchronization mechanism for individual Composed time Petri net models

any time in the interval $[0,\tau_1]$, *respectively* $[0,\tau_2]$, *the synchronization should be performed in the minimum time between* τ_1,τ_2.

As previously defined, an MITL mission is fulfilled when the place modeling the final state of the associated automaton has a token. For this example, these places are denoted: $p_1^{\varphi_1}$ *and* $p_1^{\varphi_2}$. *Since these places can be reached anytime under the time restrictions* τ_1 *and* τ_2, *then new places are added simulating the waiting action immediately after the input transitions are fired* ($t_{g_{0,1}}^{\varphi_1}, t_{g_{0,1}}^{\varphi_2}$ *which are also highlighted with an orange bounding box. Thus, once the True value of the atomic propositions is evaluated, the token modeling the current state of the specification reaches a waiting place. This approach is considered for all the robots requiring synchronizations for a set of actions.*

Once both waiting places are reached, $p_4^{\varphi_1}, p_4^{\varphi_2}$, *then a newly added transition is added* $t_{g_{0,0}}^{\varphi}$ *responsible for the synchronization of both MITL missions. The output arcs from this newly added transition are directed toward the places modeling the fulfillment of the MITL missions. In this sense, notice that the time expressed by the transition is the* $[0,0]$ *time interval, forcing an immediate firing action.* ■

Remark 5.2 The synchronization mechanism does not alter the meaning of the mission by the addition of the instantaneous transitions with time $[0,0]$. In contrast, the synchronization mechanism implies dependencies between the missions, such that one robot cannot fulfill its specifications without synchronizing with another robot.

5.3 Model-checking approach in numerical evaluation

The motion planning strategy for MITL missions is based on simulations, which solve a reachability problem commonly addressed through model-checking approaches. After computing the *Composed Time Petri net*, the model is implemented in the ROMEO tool [134], which provides a model-checking solution by determining if a particular marking can be reached. If this is the case, the sequence of transitions is provided. The marking corresponds to the place modeling the final state of the TBA model of the MITL formula. Therefore, the solution that reaches the desired marking is equivalent to satisfying the MITL formula (as demonstrated in Example 5.1.4). The paths of the multi-robotic system are then translated into motions based on the trace run and the controller, which steers the robots between partitioned cells. A notable advantage of ROMEO for modeling and analyzing Time Petri Net models is its on-the-fly model-checking procedure, which avoids the creation of the entire state-class graph during trace run searches.

Consider the case study presented in [135], where a gantry-robot system consisting of $\mathscr{R}$ robotic arms is responsible for installing reinforcement bars (rebars) to build a concrete structure (cage). This scenario involves two sub-groups of robotic arms based on their gripper capabilities: the first subgroup (denoted r_{pp}) picks up the rebars, moves them within the construction area, and holds the rebars while the second subgroup of robots, r_c, connects the rebars. The two subgroups satisfy $r_c \cap r_{pp} = \emptyset$ and $r_c \cup r_{pp} \subseteq r$. Although the solution proposed in [135] addresses this scenario, the authors note a lack of flexibility in their approach. In contrast, the current work proposes a fixed topology model regarding the number of agents in the team. For example, the number of robots assigned to each subgroup can vary depending on the type of rebars. Furthermore, time constraints are integrated into the path planning strategy. This case study builds on the assumptions in [135], where the necessary data for building the cage is known in advance.

The planning strategy is adaptable since is based on a configuration space including two regions of interest (ROIs), y_1 and y_2, related to the atomic propositions b_1, respectively b_2:

- y_1 - represents the common area associated with the set r_{pp} of robots aiming to pick up the rebars. It is considered that gripping points are allocated to the robots while ensuring collision-free movement.

- y_2 - represents the construction area where the rebars are placed by the r_{pp} robots and connected by the set of robots r_c.

Each sub-group of robots is assigned a separate MITL specification as a mission. These formulas are repeated for each rebar until the entire structure is completed. The time restrictions are defined as $[0, \tau_i], \forall I_i, \tau_i \in \mathbb{N}_+$, with both MITL specifications being interconnected through the time conditions $\tau_3 > \tau_4$ and $\tau_3 \leq \tau_4 + \tau_5$ (where the robots in r_{pp} must hold the rebars for a duration of $\tau_5 = |I_3|$, while the robots in r_c connect the rebars).

(i) Formula (5.1) is allocated to the robotic set r_{pp}, mentioning the sequence of actions: picking up rebar from region y_1 under the time restriction from interval I_1, placing it in region y_2 within a time interval I_2 relative to the pick-up time, and holding the rebar for a duration of I_3.

$$\varphi_{r_{pp}} = \Diamond_{I_1} b_1 \wedge (b_1 \rightarrow \Diamond_{I_2} \Box_{I_3} b_2) \tag{5.1}$$

(ii) Formula (5.2) demand the robots r_c to visit region y_2 within time I_4, while remaining there for I_5 units of time to connect the rebars together.

$$\varphi_{r_c} = \Diamond_{I_4} \Box_{I_5} b_2 \tag{5.2}$$

The core idea is to build a separate *Composed Time Petri net* for each MITL specification and use a model-checking approach to assess whether the specification can be satisfied based on the $\mathscr{TPN}^C$ representation.

Algorithm 9: Motion of the entire robotic team

Input : r, $\varphi_{r_{pp}}$, φ_{r_c}, $structure_{fixed}$, E
Output : Planned robotic motion

Initially place all the robots r in their home position;
$structure = \emptyset$;
Construct $\mathscr{TPN}^{\varphi_{r_{pp}}}$ based on mission $\varphi_{r_{pp}}$;
Construct $\mathscr{TPN}^{\varphi_{r_c}}$ based on mission φ_{r_c};
while $structure \neq structure_{fixed}$ **do**
    A rebar arrives at the pick-up region y_1;
    Evaluate rebar's type to count the number of agents in sub-groups r_{pp} and r_c;
    Determine and follow paths for setsr_{pp} and r_c;
    Update $structure$;
    Move r_c and r_{pp} in their initial home position;
end

Algorithm 9 outlines the procedure followed in this case study. Initially, all robots are positioned in their home location. For each rebar arriving at the pick-up area y_1, the number of robots assigned to each subgroup is calculated based on the type of rebar, as described in [135]. Each subgroup, r_{pp} and r_c, is assigned a separate MITL formula, and an individual $\mathscr{TPN}^C$ model is computed for each. These formulas are satisfied when robots r_{pp} fulfill $\varphi_{r_{pp}}$, and robots r_c satisfy φ_{r_c}. This corresponds to reaching the final markings in the TPN models, with y_i being true only when the token count equals $|r_{pp}|$ for the first subgroup, and $|r_c|$ for the second. After the transition sequences in the TPN models (representing robot paths) are determined, the robots return to their home position, and the structure is updated.

The process from lines 6 to 13 is repeated until the current structure matches the intended structure, $structure_{fixed}$, provided as input.

Table. 5.1. Numerical Evaluation of the *Composed Time Petri net*

MITL Formula	**Size of $\mathscr{TPN}^C$**	**No. of agents**	**Model-checking run time [sec]**
$\varphi_{r_{pp}}$	$\lvert P^C\rvert = 50, \lvert T^C\rvert = 65$	2 3	2.2 81.3
φ_{r_c}	$\lvert P^C\rvert = 42, \lvert T^C\rvert = 58$	2 3	1.8 49

Consider the workspace partitioned into cells, represented by $\mathscr{TPN}^E$ with 20 places. Table 5.1 presents the numerical evaluation for the MITL specifications (5.1) and (5.2), reflecting the size of each *Composed Time Petri net* model $\mathscr{TPN}^C$, specifically $\mathscr{TPN}^{\varphi_{r_{pp}}}$ and $\mathscr{TPN}^{\varphi_{r_c}}$. The simulation results were computed on a computer with an i7 8th-generation CPU @ 2.20GHz and 8GB RAM.

Remark 5.3 As stated in [100], both the TBA and TPN models are timed bisimilar. If each robot from the multi-robotic team is modeled as a TBA, as in [39], the total number of nodes for the entire team increases exponentially due to the product automata, which also includes the places for the TBA assigned to an MITL formula φ. In contrast, the size of a *Composed Time Petri net* model in terms of the number of places P^C is determined by the cardinality of the sets $P^E, P^O, P^{\neg O}$, and P^φ. Importantly, the total number of places P^C is not influenced by the number of robots involved in the $\mathscr{TPN}^C$.

Considering the tool ROMEO, a marking that can be reached leads to a sequence of transitions. The tool uses TCTL (Timed Computation Tree Logic) for this evaluation. For instance, to compute the robotic trajectories ensuring the MITL mission $\varphi_{r_{pp}}$, the following TCTL formula is applied for model-checking: $EF[0,c_u](M(p_f^C) == 1)$, where $c_u \in \mathbb{N}_+$ is a user-defined time constant. In simple terms, this TCTL formula indicates that if a trace eventually reaches the desired marking modeled by p_f^C, the sequence of transitions is returned. Otherwise, the formula cannot be fulfilled.

Discussion. The proposed TPN structure effectively coordinates the local time constraints associated with robot motion ($\mathscr{TPN}^E$) and the global time constraints defined by the MITL specification ($\mathscr{TPN}^\varphi$). Furthermore, the *Composed Time Petri net* model is suitable for representing the census concept, as discussed in [132], which requires multiple agents to complete a task.

The model-checking approach applied in this case study demonstrates the effectiveness of the proposed framework, which offers a fixed topology model relative to the number of agents needed to satisfy an MITL formula. Thus, this work based on t=Time Petri nets provides a solid foundation for developing scalable multi-agent system models. However, one

limitation of the current approach arises when handling a large number of robots. Although the ROMEO model-checking tool does not explore the entire reachability graph, simulations involving more than 3 agents failed to provide a solution. To address this, alternative approaches such as structural methods (e.g., mathematical programming) or techniques that partially explore the state space (e.g., distributed methods) could be further explored.

Chapter 6

Path planning with LTL specification based on hierarchical approach for multi-robot system

This chapter addresses the motion planning problem for heterogeneous teams of robots ensuring a global specification. The planning strategy involves a hierarchical structure of Petri net models, known as the *Nets-within-Nets* (NwN) paradigm [57]. In this sense, a novel framework is proposed, denoted *High-Level robot team Petri Net* (HLrtPN) system having two characteristics. Firstly, the robots' behavior, alongside a model of the team's mission, is represented by a set of *object nets*. Secondly, a *system net* is defined to provide the global state of the robotic system, by coordinating the object nets through *Global Enabling Function* (GEF). The solution is obtained by simulating the HLrtPN system using specialized software designed to support Nets-within-Nets (NwN). Demonstrative examples, based on Linear Temporal Logic (LTL) missions, highlight the computational feasibility of the proposed framework, accompanied by numerical analyses across various DES representations. A review of the existing literature suggests that this approach is among the first to present a step-by-step motion planning solution leveraging the NwN paradigm.

The contributions included in this chapter are as follows:

- Introducing a novel framework, the *High-Level robot team Petri Net* (HLrtPN) system, for motion planning in heterogeneous robotic systems to guarantee the fulfillment of a global mission. To achieve this, a synchronization function, referred to as the *Global Enabling Function*, is developed. This function is responsible for verifying and executing a set of logical Boolean formulas to ensure compliance with the specified requirements.

- Describing the step-by-step implementation of the framework in Renew [136] and making it accessible on web [137]. The illustrative examples of this implementation

showcasing the modeling of LTL formulas for heterogeneous robotic teams strengthen our framework's substantial potential in robotic planning;

- Assessing the proposed solution through numerical simulation in Renew and comparing it with various DES representations, considering two case studies, one of which includes a real-life futuristic scenario for a robotic team evolving in a hospital (detailed in Chapter 7.

6.1 Nets-within-Nets paradigm

Recently, motion planning in the robotic field represents a challenging problem to be solved, considering daily applications where the multi-robotic systems are heterogeneous, e.g., agriculture [138] and mapping [139]. As mentioned also in Chapter 1, the objective of this thesis is to provide planning strategies while preserving the advantages of discrete-event-based frameworks. In Chapter 2.1 is stated that the Petri net formalism facilitates an easier modeling of homogeneous robotic teams, by associating a token to each robot. Accounting for heterogeneous robots, the use of the Petri net model defined in Definition 2.2.2 can represent a barrier. One might suggest the use of different classes of PN, e.g., colored Petri nets [140]. Another perspective includes the coordination of multiple PNs in a structured manner.

The solution described in this chapter is based on a hierarchical approach of Petri net models, known as the Nets-within-Nets paradigm [57]. The particularity of this family of high-level nets is characterized by the fact that each token can transfer information such as states of another process. In this sense, a token is visualized as a Petri net denoted as *Object net*. Moreover, the relation between these nets is captured in *System net*, which contains a global view of the entire system [58]. The object-oriented methodology is suitable for high-level representations [58] by introducing different types of mobility among nets [141], which express synchronous and synchronous actions.

Example 6.1.1 *Let us provide a short illustration based on which the fundamentals of the Nets-within-Nets paradigm are explained, such as system net and object net. In Figures 6.1 - 6.3, it can be observed the system net, given by places p_3, p_4 and transition t_3, and the object net given by places p_1, p_2 and transitions t_1, t_2.*

This hierarchical structure of nets allows the paradigm to capture three different behaviors. Figure 6.1 portrays the transport of the object net into the system net. Specifically, the marking of the object net is not modified by firing the transition t_3. Figure 6.2 presents the autonomous transition behavior, representing the marking's update only in the object net, by firing only transition t_1. In Figure 6.3, an interaction between both system net and object net is portrayed, through the synchronous firing of two transitions t_1 and t_3.

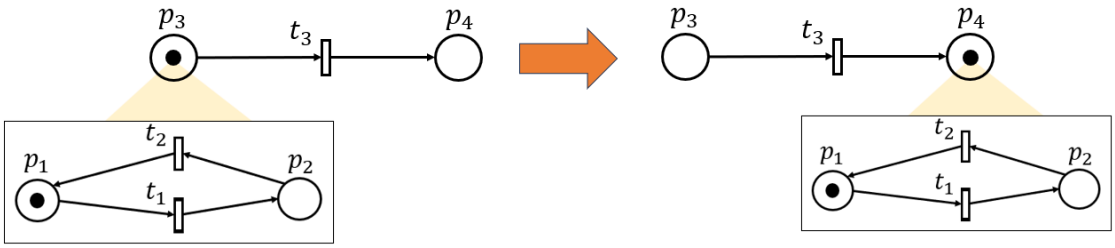


Fig. 6.1. Nets-within-Nets: transport

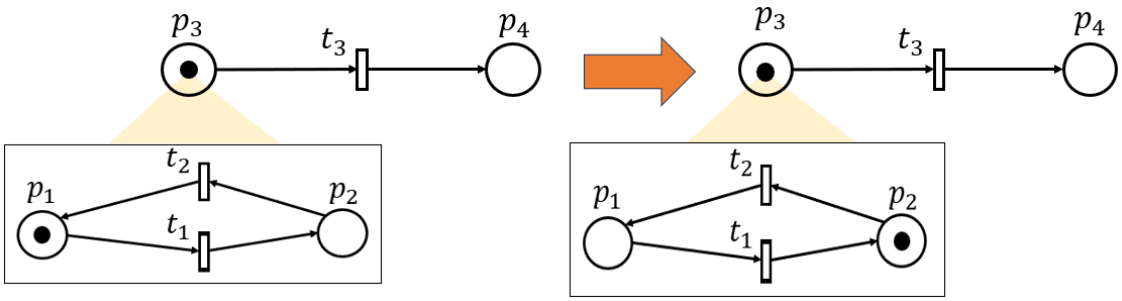


Fig. 6.2. Nets-within-Nets: autonomous transition

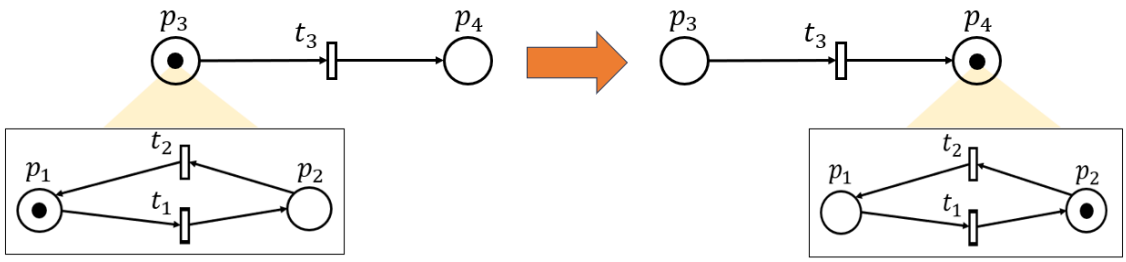


Fig. 6.3. Nets-within-Nets: interaction

For an easier understanding, explanations from the manufacturing domain shall be provided for each behavior. For example, the transport action updates the global view of the system, e.g., the movement of a robot from p_3 to p_4, while maintaining the local state of a robot, e.g., gripper closed (given by the presence of a token in p_1); the autonomous transitions illustrates update only for local states, e.g., from gripper closed (marking in p_1) to gripped open (marking in p_2) while maintaining the global state, e.g., robot doesn't move (marking in p_3); and the interaction action which synchronously updates both the global and local states of the entire system, e.g., robot moving from p_3 to p_4 while opening the gripper (marking in p_2) from a close state (marking in p_1). ■

Remark 6.1 Generally, there is only one system net that provides the global state of a system. On the other hand, for each sub-system, the states are given by individual object nets, which can have various designs, e.g., two different robots modeled by two different object nets representations.

Several application examples of the NwN paradigm include modeling web service coordination [142], smart houses [143], modeling the epigenetic regulation process at the cell level [144], and simulating antibiotic resistance at the microbiota level [145]. Other works focused on self-development tools such as *Renew* [136] and *Modular Model Checker* (MoMoC) [146], or encoding specifications in *Maude language* [147].

When writing this thesis, the total number of papers that involve the "Nets-within-Nets paradigm" is 64, using the Web of Science database. Of this number, only a few papers address the problem of computing paths for single or multiple robots [141]. Some papers use the object-oriented or hierarchical idea of PNs, without referring to it as the NwN paradigm [148]. This work introduces a top-down framework with formal definitions under the NwN paradigm, suitable for heterogeneous robotic teams that ensure a global specification. In this account, the team's mission is also modeled by an object net, interacting further with object nets allocated to the robots through a synchronization function that secures spatial capacity constraints by design (the maximum number of robots in an environmental region). The following contributions can be mentioned:

- Proposing a novel framework called the *High-Level robot team Petri Net* (HLrtPN) system for motion planning of heterogeneous robotic systems that ensures a global mission. For this purpose, a synchronization function (*Global Enabling Function*) between the nets is designed, having the role of verifying and acting on a set of logical Boolean formulas to ensure that the specification requirements are met;

- Describing the step-by-step implementation of the framework in Renew [136] and making it accessible on web [137]. The illustrative examples of this implementation showcasing the modeling of LTL formulas for heterogeneous robotic teams strengthen our framework's substantial potential in robotic planning;

- Assessing the proposed solution through numerical simulation in Renew and comparing it with various DES representations, considering two case studies, one of which includes a real-life futuristic scenario for a robotic team evolving in a hospital.

Currently, the framework HLrtPN facilitates an easier modeling approach in the robotic field, but the drawback is that the planning strategy leading to a feasible solution requires the exploration of multiple transitions for complex systems, which can be time-consuming. Nevertheless, the potential of HLrtPN can be further exploited in search of viable trajectories of a heterogeneous robotic team ensuring a global mission.

6.2 Problem formulation

Problem: This work addresses the task allocation and planning problem for a heterogeneous multi-robotic system evolving in a known environment including a set of regions of interest. The team should ensure a global mission given as a co-safe LTL specification, imposing spatial constraints of visiting/avoiding the regions, and temporal constraints requiring sequencing and synchronization.

The solution space of the problem is subject to a proposed framework under the Nets-within-Nets formalism correlating hierarchical Petri net representations, including both the motion of robots and the global mission to determine a solution that will be simulation-based.

An explanation of the proposed method consists of modeling the allowed movements of the heterogeneous team as a set of PNs (one assigned to each different type of robot, as sketched in Figure 6.4-iii), also specifying the mission in the same formalism (as depicted through the PN in Figure 6.4-i). These models are implemented at the same hierarchical level as *Object nets*. The upper-level PN denoted *System net*, considers object nets as tokens (Figure 6.4-ii). Firing the transitions of the system net imposes synchronization between the object nets (of the robots and the mission).

The prerequisites of the problem formulation include the following assumptions:

- Among the high-level languages to specify a mission for the robotic team, we have focused on the Linear Temporal Logic formalism, due to a provided algorithm from our previous work in [35], which allows us to model the Büchi automaton for an LTL formula into a Petri net model. Additional details are provided in Chapter6.3.1.

- The high-level planning of mobile robots returned by the proposed method is composed of a sequencing of motion plans which a low-level controller of the robots can enforce. Thus, robots are not restricted to any particular category, encapsulating various dynamic characteristics.

An intuitive explanation is provided for a clearer understanding of this concept. The NwN model is called the *High-Level Robot Team Petri Net* (HLrtPN) system, and it comprises

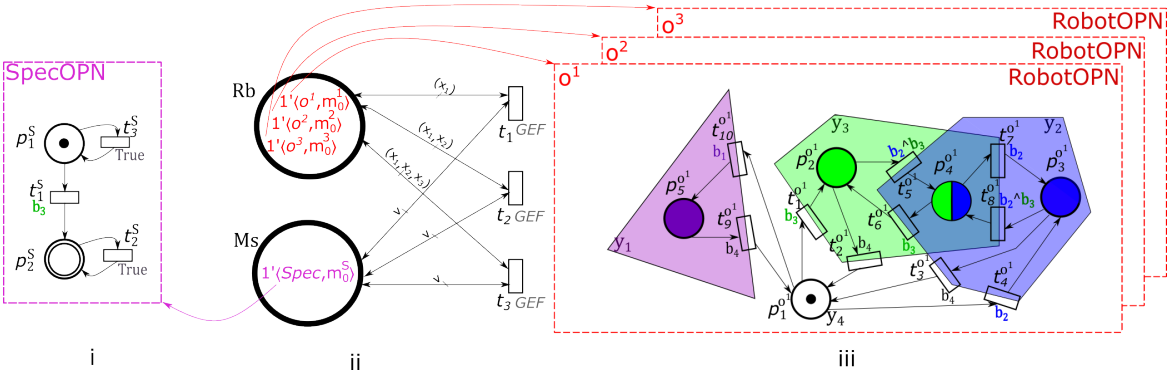


Fig. 6.4. Example demonstrating the Nets-within-Nets paradigm: (i) Specification Object Petri net, (ii) System net, (iii) Robotic Object Petri net

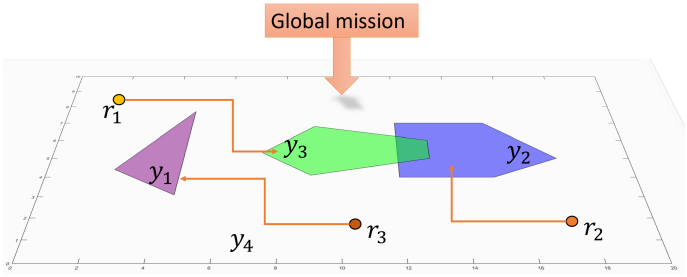


Fig. 6.5. Example of an environment with 4 ROIs and 3 robots initially placed in the free space y_4 and having the trajectories for the mission $\varphi = \Diamond b_1 \wedge \Diamond b_2 \wedge \Diamond b_3 \wedge (\neg b_1 \mathscr{U} b_3)$ (meaning to visit y_1, y_2, y_3, with y_3 before y_1)

(i) a set of object nets modeling robots (*Robotic Object Petri Nets* (RobotOPNs)) and one object net modeling the mission (*Specification Object Petri Net* (SpecOPN)); and (ii) one system net where each token corresponds to an object net. The *system net* governs the evolution of the system. When transitions of the RobotOPNs are fired, they must fire synchronously with a transition in the system net. Additionally, when transitions in the RobotOPN are fired, the robots move between regions, updating the values of set atomic propositions $\mathscr{B}$. Consequently, the transition fired in the system net synchronizes with the firing of a transition in SpecOPN. The overall synchronization of the transitions in the system and object nets is ensured by the synchronization function *GEF*.

Let us consider the environment in Figure 6.5 that can be partitioned into 5 cells $\mathscr{C} = \{c_1, \ldots, c_5\}$ where c_1, c_2, c_3, c_4, respectively c_5 are associated with the free space (white region), green region minus the intersection with blue region; the blue region minus the intersection with the green region; the intersection of the blue and green region, respectively the purple region. The set of labels is b_1 for the purple region y_1, b_2 for the blue region y_2, b_3 for the green region y_3, and b_4 for the free space, which is represented by region y_4. Therefore, the labeling function h is defined as follows: $h(c_1) = \{b_4\}$, $h(c_2) = \{b_3\}$,

$h(c_3) = \{b_2\}$, $h(c_4) = \{b_2, b_3\}$ ($h(c_4) = b_2 \wedge b_3$) and $h(c_5) = \{b_1\}$. Note that for this intuitive explanation, the labeling function h is defined on the set of cells $\mathscr{C}$. This function shall be defined further on the set of places P modeling the RobotOPN representations, considering that each place is associated with a cell $c \in \mathscr{C}$.

The HLrtPN is illustrated in Figure 6.4. Specifically, Figure 6.4-ii displays the system net with two places: *Rb*, containing three tokens, each corresponding to a RobotOPN of a robot, and *Ms*, with one token corresponding to the SpecOPN. The system net includes three transitions: t_1, t_2, respectively t_3, which fires only one robot, two robots, and three robots changing their regions.

Remark 6.2 Formally, to differentiate between the object nets, we have defined the following notations such as superscripts: "S" for SpecOPN, respectively "o^i" for RobotOPN modeling the robot r_i, for all the components as part of these nets definition (Chapter6.3.1).

Figure6.4-i shows the SpecOPN for the simple formula $\varphi = \Diamond b_3$. The mission is assumed to be accomplished when p_2^S has a token (or a robot reaches the region y_3). Notice that this is possible by the firing transition t_1^S attached to the labeling function value b_3. The function is evaluated at $True$ when a robot enters a region y_3 (cells c_2 or c_4).

Figure 6.4-iii shows one RobotOPN denoted as o^1. The other two are identical if the other two robots are identical. In the RobotOPN of robot r_j, each place models a cell from $\mathscr{C}$; specifically, for each $c_i \in \mathscr{C}$, a place $p_i^{o^j}$ is defined, where o^j models the robot r_j. Initially, all robots are in the free space, with the initial marking of o^j, $j = \overline{1,3}$, being a token in $p_1^{o^j}$. Since the label of c_1 is $h(c_1) = \{b_4\}$ (the region y_4 modeled by $p_1^{o^j}$), the atomic proposition b_4 is evaluated as $True$ in this state. If robot r_1 leaves c_1 and enters c_2, transition $t_1^{o^1}$ should fire, and the atomic proposition b_3 becomes $True$ since $h(c_2) = \{b_3\}$ (for region y_3). Notice that the transitions in RobotOPN are labeled with the atomic propositions evaluated as $True$. The movement of a robot from one cell to another updates the truth value of at least one atomic proposition. Adjacent cells have different labels but our modeling methodology can handle more general cases.

Suppose we return to the firing of transition $t_1^{o^1}$, assuming that the other robots are not moving. In that case, this means that transition t_1 from the system net should fire synchronously with $t_1^{o^1}$ (in the system net, transition t_1 models the movement of one robot, t_2 the synchronous movement of two robots, and t_3 the synchronous movement of all three robots). Moreover, the defined GEF will ensure the synchronous firing of these two transitions with one transition from the SpecOPN. In SpecOPN, both transitions are enabled and the logical functions assigned to them are also true (t_3^S is labeled with $True$ while t_1^S is labeled with b_3, which will become $True$ as the robot enters the green region). If transition t_1^S fires, p_2^S will have a token and the mission will be fulfilled.

Remark 6.3 The simulation results highlight the benefits of the current method when compared with the other three DES approaches (two of them being restricted to homogeneous teams), one rational being represented by the versatility of the Nets-within-Nets formalism

of modeling heterogeneous robotic teams. When a heterogeneous robot is added to the team, it is necessary to add a RobotOPN model only, allowing for easier handling of the entire framework in comparison with traditional approaches based on transition systems.

The framework is validated through numerical experiments using simulations performed in specialized software tools. Consequently, the current approach yields a sub-optimal solution instead of pursuing an optimal one by exhaustively exploring the reachability graph of different models or solving intricate optimization problems. A key advantage of the proposed framework lies in its ability to establish tailored connections between robots and specific tasks, making it well-suited for addressing complex scenarios. Additionally, it has the potential to incorporate time-analysis mission models from other formalisms to enhance its applicability.

6.3 Nets-within-Nets tailored to path planning

The proposed framework *High-Level Robot Team Petri Net* (HLrtPN) is defined formally below, including definitions and illustrative examples for a clear understanding.

6.3.1 Object Petri nets systems

The dynamic of the heterogeneous robotic team is modeled by a set of object nets *Robotic Object Petri net* (RobotOPN), one assigned to each type of robot based on their spatial capabilities (allowed ROIs to reach). Respectively, another object net called *Specification Object Petri net* (SpecOPN) models the requirements included in the global mission, which the team should ensure. The following subsections formally define these nets.

Specification Object Petri net

Definition 6.3.1 (SpecOPN) *A* Specification Object Petri net *represented by a tuple* $Spec = \langle P, P_f, T, F, \lambda \rangle$*, where:* P *and* T *are the disjoint finite set of places and transitions,* $P_f \subseteq P$ *is the set of final places,* $F \subseteq (P \times T) \cup (T \times P)$ *is the set of arcs. The transition labeling function* $\lambda_\wedge(t) \equiv t_\wedge$ *assigns to each transition* $t \in T$ *a Boolean formula defined by using the atomic propositions* $\mathscr{B}$ *and their negations.* ■

The current work considers SpecOPN the translated PN model obtained from the Büchi automaton (Algorithm 5, Chapter 4). This translation accounts only for the transitions considered in the Büchi automaton, without generating new ones, as in Composed Petri net model, for which the generation of virtual transitions was necessary for the planning strategy. The translated Petri net is a *state machine* PN, and at any time, only one place is marked. Moreover, the proposed translation from the Büchi automaton adapts any disjunctive Boolean

formula from the automaton $b_i \vee b_j$ to a conjunctive Boolean formula, thus returning two transitions with their labeling functions $\lambda_\wedge(t_x) = b_i$ and $\lambda_\wedge(t_z) = b_j$.

A marking is represented as a $\{0,1\}$-valued vector of size $|P|$, while a SpecOPN system is defined as the pair $\langle Spec, m_0 \rangle$, where m_0 denotes the initial marking. The specification is satisfied when the SpecOPN system reaches a marking with a token in a place belonging to P_f, achieved by firing a sequence of enabled transitions. A transition $t \in T$ within the SpecOPN system is enabled at a marking m if two conditions are satisfied: (i) $m[{}^\bullet t] = 1$, [1] and (ii) the Boolean formula $t_\wedge$ evaluates to *True*. Informally, condition (i) serves as the enabling condition, while condition (ii) implies that the movement of robots within the set of regions $\mathscr{Y}$ triggers the firing of a transition in SpecOPN by altering the truth value of $t_\wedge$. In Figure 6.4-i, the final place is p_2^S, which becomes marked only when t_1^S fires. This occurs when a robot enters a region labeled with b_3.

Robotic Object Petri net

It is assumed that RobotOPN is a state machine PN that can be considered an *labeled Petri net* [149] by adding a labeling function over the set of transitions and places.

Definition 6.3.2 (RobotOPN) *Given a set of cells $\mathscr{C}$ modeling the workspace of the robotic team, let the* Robotic Object Petri net *be the model of a robot, expressed by the tuple $o = \langle P, T, F, h, \lambda, \gamma \rangle$:*

- *P is the finite set of places, bijective with set $\mathscr{C}$. Each place is associated with an element $c_i \in \mathscr{C}$ in which the robot is allowed to enter;*
- *T is the finite set of transitions. A transition $t_{ij} \in T$ is added between two places $p_i, p_j \in P$ only if the robot can move from any position in cell c_i to cell c_j in the partitioned workspace E;*
- *$F \subseteq (P \times T) \cup (T \times P)$ is the set of arcs. If t_{ij} is the transition modeling the movement from p_i to p_j, then $(p_i, t_{ij}) \in F$ and $(t_{ij}, p_j) \in F$;*
- *$h_\wedge$ is the labeling function of places $p \in P$, defined in the previous section and associating to each place a Boolean formula over the set of propositions $\mathscr{B}$;*
- *$\lambda_\wedge$ is the Boolean labeling function of transitions $t \in T$, such that $\lambda_\wedge(t_i) = h(t_i^\bullet)_\wedge$;*
- *$\gamma : P \to \mathscr{C}$ is the associating function. If place $p_i \in P$ is associated with $c_i \in \mathscr{C}$, then $\gamma(p_i) = c_i$.* ■

[1] ${}^\bullet t$ and $t^\bullet$ represent the input and output places of transition $t \in T$, respectively, which are singletons since the SpecOPN is a state machine.

The marking of the RobotOPN is a vector $m \in \{0,1\}^{|P|}$. Initial marking is denoted m_0 such that $m_0[p_i] = 1$ if the robot is initially in p_i, and $m_0[p_j] = 0$ for the rest of the places $p_j \in P \setminus \{p_i\}$. A RobotOPN system is a pair $\langle o, m_0 \rangle$.

A heterogeneous robotic system incorporates the dynamics of several types of robots. We are interested in the differences concerning their space constraints (ROIs that can be reached). Each type of robot is modeled as a RobotOPN, including these differences in terms of topology and labels.

6.3.2 High-Level robot team Petri net

This section introduces our proposed model denoted *High-Level robot team Petri net* which encapsulates the ability to provide a global view, by enabling synchronizations between the system net and the previously defined object nets.

Definition 6.3.3 *A* High-Level robot team Petri net *(HLrtPN) is a tuple* $\mathscr{N} = \langle \bar{P}, \bar{T}, \mathscr{O}, \mathscr{S}, Vars, \bar{F}, W, \mu_{cap} \rangle$*, where:*

- $\bar{P} = \{Rb, Ms\}$ *is the set of places;*
- $\bar{T} = \{t_1, t_2, \ldots, t_s\}$ *is the set of transitions;*
- $\mathscr{O} = \{\langle o^1, m_0^1 \rangle, \langle o^2, m_0^2 \rangle, \ldots, \langle o^n, m_0^n \rangle\}$ *is a set of n RobotOPN systems, one for each robot;*
- $\mathscr{S} = \langle Spec, m_0^S \rangle$ *is a SpecOPN system;*
- $Vars = \{v, x_1, x_2, \ldots, x_n\}$ *is a set of* variables*;*
- $\bar{F}$ *is the set of arcs:* $\bar{F} = \bigcup_{t \in \bar{T}, p \in \bar{P}} \{(p,t),(t,p)\}$*;*
- W *is the inscription function that assigns a set of variables from Vars to each arc. For every* $t_i \in \bar{T}$*, it holds that* $W(Rb,t_i) = W(t_i,Rb) = (x_1, x_2, \ldots, x_i)$ *and* $W(Ms,t_i) = W(t_i,Ms) = v$*.*
- $\mu_{cap} \in Bag(\mathscr{P})$ *represents the* capacity multi-set*, where* $\mu_{cap}[P_i] > 0$ *for all* $i \in \{1, \ldots, w\}$*, and* $\mu_{cap}[P_j] \geq n$ *if* $h(P_j) = b_q$*.* ■

The system net (as noticed in Figure 6.4-ii.) is the tuple $\langle \bar{P}, \bar{T}, \bar{F}, Vars, W \rangle$, with Rb and Ms being the robot and mission places. The transitions are connected via bidirectional arcs, where t_i synchronizes i robots according to the specification, with $i = \overline{1,s}$. Considering co-safe LTL missions, the following assumption is made: $s \leq n$. The firing of a transition manipulates the object nets through the use of variables, e.g., x_1 is bound to a RobotOPN, v for the SpecOPN. Although the state of an object net changes when a transition in the HLrtPN system is fired, the reference semantics approach, as described in [58], ensures that

a token serves as a reference to an object net. The same variable is used bidirectionally—for both input to and output from places and transitions.

Each cell $c_i \in \mathscr{C}$ has a specific number of space units, referred to as its *capacity*, denoted by $\mu_{cap}[c_i]$. This value is determined by the multi-set μ_{cap} and represents the maximum number of robots that can occupy cell c_i at the same time. It is assumed that each robot occupies one unit of capacity. Consequently, every cell $c_i \in \mathscr{C}$ has a strictly positive capacity, with the additional assumption that a cell $c_j \in \mathscr{C}$ labeled as $h(c_j) = b_q$ has sufficient space to accommodate the entire team (as described in the last bullet of Def. 6.3.3).

An HLrtPN system is defined as a tuple $\langle \mathscr{N}, m, \mu_{occ} \rangle$, where $\mathscr{N}$ represents an HLrtPN as in Definition 6.3.3, m denotes the marking that associates a multi-set to each place in $\bar{P}$, and μ represents the multi-set. The marking is represented as a multi-set that assigns a non-negative integer coefficient to each element. The set of all multi-sets over U is denoted as $Bag(U)$. The algebra of multi-sets, as defined in [150], includes various operations such as addition and comparison.

The initial marking m_0 is

- $m_0[R_b] = 1'\langle o^1, m_0^1 \rangle + 1'\langle o^2, m_0^2 \rangle + \ldots + 1'\langle o^n, m_0^n \rangle$;
- $m_0[Ms] = 1'\langle Spec, m_0^S \rangle$.

Finally, $\mu_{occ} \in Bag(\mathscr{B})$ represents the *occupancy multi-set*, which indicates the current positions of the robots relative to the regions in $\mathscr{Y}$. At any given moment, $\mu_{occ}[b_i]$ denotes the number of robots currently present in the region y_i. The initial occupancy multi-set is defined as $\mu_{occ_0} = \sum_{i=1}^{q-1} 0'b_i + \sum_{i=1}^{n} |R_k|'b_j$, assuming that all robots are initially located in the free space.

A transition $t \in \bar{T}$ of the HLrtPN is *enabled* at a given state $\langle m, \mu_{occ} \rangle$ if the following conditions are satisfied:

- $m[Ms]$ has an enabled transition $t^S \in T^S$;
- Given $W(Rb,t) = (x_1, x_2, \ldots, x_i)$, $m[Rb]$ contains i RobotOPN net systems ($\langle o^1, m^1 \rangle, \langle o^2, m^2 \rangle, \ldots, \langle o^i, m^i \rangle$), where each net has an enabled transition t^{o^j}, for $j = \overline{1,i}$, and $GEF(\mu_{occ}, \mu_{cap}, t^S, (t^{o^1}, t^{o^2}, \ldots, t^{o^i})) = \text{True}$.

An enabled transition $t \in \bar{T}$ may fire, transitioning the system from $\langle m, \mu_{occ} \rangle$ to $\langle m', \mu'_{occ} \rangle$, such that:

- $m'[Ms]$ include the firing of transition t^S;
- At $m'[Rb]$, each o^i has fired its corresponding transition t^{o^i};
- μ'_{occ} is updated to reflect the new positions of the robots.

6.3.3 Synchronization function

When a transition $t_j \in \bar{T}$ of the HLrtPN fires, the system should synchronize transitions in both the RobotOPNs (from $m[Rb]$) and SpecOPN (from $m[Mb]$). However, this synchronization is subject to various compatibility rules that consider the current state of the system, including the occupancy multi-set μ_{occ}. To guarantee these rules are ensured, the *Global Enabling Function* (GEF) acts as a gatekeeper, checking the compatibility of the system's state involving the transition rules before enabling synchronous transitions. The GEF is essential for ensuring that the firing of an HLrtPN transition, together with the corresponding enabled transitions in RobotOPNs and SpecOPN, occurs without violating any system rules.

For any transition $t \in \bar{T}$, the GEF takes inputs from $Bag(\mathscr{C}) \times Bag(\mathscr{B}) \times T^S \times \left(\bigcup_{i=1}^{n} \prod_{j=1}^{i} T_j^k\right)$ and returns either $True$ or $False$ to enable or disable t. The GEF evaluates the assignment of variables to input arcs (i.e., v and $(x_1, x_2, \ldots, x_i)$), along with global data such as the occupancy multi-set μ_{occ}, the capacity multi-set μ_{cap}, the marking-enabled transition t^S in the SpecOPN, the Boolean label v, and the set of marking-enabled transitions $(t^{o^1}, t^{o^2}, \ldots, t^{o^i})$ in $(x_1, x_2, \ldots, x_i)$. If firing the i transitions in the RobotOPN satisfies the Boolean label for t^S, the GEF returns $True$; otherwise, it returns $False$. The algorithm for this function is provided in Algorithm 10.

Algorithm 10: The Global Enabling Function (*GEF*)

Input: $\mu_{occ}, \mu_{cap}, t^S, (t^{o^1}, t^{o^2}, \ldots t^{o^i})$
Output: Is the synchronized firing of $t^S, t^{o^1}, t^{o^2}, \ldots, t^{o^i}$ feasible?
Data: $(\langle o^1, m^1 \rangle, \langle o^2, m^2 \rangle, \ldots, \langle o^n, m^n \rangle), \mathscr{C}$

Let χ be the simulated occupancy multi-set w.r.t. $\mathscr{C}$ after firing $(t^{o^1}, t^{o^2}, \ldots t^{o^i})$;
forall $c_j \in \mathscr{C}$ **do**
    if $\left(\chi[c_j] > \mu_{cap}[c_j]\right)$ `/* See Comment 1 */`
    then
        return *False*
if $\left(t^S_{\wedge} == True\right)$ `/* See Comment 2 */`
then
    return *True*
else
    Let μ'_{occ} be the simulated update of μ_{occ} w.r.t. $\mathscr{B}$ after firing $(t^{o^1}, t^{o^2}, \ldots t^{o^i})$;
    forall $\left(b_j \in \mathscr{B}\right)$ `/* See Comment 3 */`
    do
        if $\left(b_j \in t^S_{\wedge} \wedge \mu'_{occ}[b_j] == 0\right) \vee \left(\neg b_j \in t^S_{\wedge} \wedge \mu'_{occ}[b_j] \geq 1\right)$ **then**
            return *False*
    return *True*

Comment 1: The enabling of a transition $t \in \bar{T}$ is checked by simulating the firing of the corresponding i transitions in the RobotOPN, which are synchronized through transition t_i of the HLrtPN. This simulation is performed using the multi-set χ, which is computed at line 1. To compute χ, transitions $(t^{o^1}, t^{o^2}, \ldots, t^{o^i})$ are fictitiously fired in the corresponding RobotOPNs (from o^1 to o^i), while no transitions are fired in the remaining RobotOPNs (from o^{i+1} to o^n). As a result, the marked places in all RobotOPNs are taken into account. By using the associating function γ^k for each RobotOPN, the multi-set χ is obtained. The *GEF* then verifies whether the firing of the transitions satisfies the capacity constraints for each $c_j \in \mathscr{C}$ (lines 2-5).

Comment 2: If the capacity restrictions are ensured and the Boolean formula assigned to t^S (i.e., $t^S_\wedge$) evaluates to *True*, the transitions $\langle t^S, (t^{o^1}, t^{o^2}, \ldots, t^{o^i})\rangle$ can fire synchronously without the need to evaluate the robot positions. In this case, the *GEF* returns *True* (line 8).

Comment 3: If the conditions are not met, a new simulation is performed, and the updated occupancy multi-set μ'_{occ} is computed. Note that χ and μ'_{occ} represent the simulated occupancy multi-sets with respect to $\mathscr{C}$ and $\mathscr{B}$, respectively. For μ'_{occ}, there are two additional conditions that may prevent the transitions from firing. These conditions are checked at lines 11-14 and are as follows:

- If an atomic proposition $b_j \in \mathscr{B}$ is part of the formula $t^S_\wedge$, but in the simulated occupancy state μ'_{occ} (after the transitions are fired), no robot is present in region y_j, then the motion of the robotic team does not fulfill the Boolean function assigned to transition t^S (first condition at line 13).

- If a negated atomic proposition $\neg b_j$ (where $b_j \in \mathscr{B}$) is part of the formula $t^S_\wedge$, but the artificial updated occupancy multi-set μ'_{occ} after the firing of the involved transitions respects $\mu'_{occ}[b_j] \geq 1$, meaning at least one robot is in region y_j, then the formula $t^S_\wedge$ is not satisfied (second condition at line 13).

If any of these conditions hold, the GEF returns *False* (line 14). Otherwise, it returns *True* (line 15). The firing of transition t (lines 8 and 15) updates the system's marking and the multi-set μ_{occ}.

Remark 6.4 A path planning solution for the proposed system HLrtPN is returned only if the capacities constraints of the robots over a set of cells $\mathscr{C}$ mapped to the set $\mathscr{Y}$ are considered such that the mission can be accomplished. A counterexample for which the executability of our model produces a deadlock is represented by the LTL mission $\varphi = \Diamond b_1$ requiring the visit of the region y_1 for which no robot can enter due to its size. This characteristic translates into $\mu_{cap}[c_i] < R - k, c_i \in \mathscr{C}$ for any robot R_k able to enter any c_i with $h(c_i) = b_1$ (for y_1). Throughout the simulations provided in the next section, we have considered feasible missions that could be satisfied by the robotic team. Thus, we have excluded scenarios that could lead to blocked or endless simulations, as previously stated.

6.4 Numerical evaluation

The evaluation of the HLrtPN system considers a detailed explanation of an easy-to-follow example, showcasing the unique perspective of the proposed model for a planning strategy of a heterogeneous robotic team. The implementation relies on the development process in Renew software tool[2][136]. The version of the software is 4.1, since this version simplifies compilation and simulation based on synchronous channels.

The feasibility of the LTL formulas addresses the number of robots in the team and their spatial constraints. A SpecOPN model associated with an LTL formula can be generated automatically based on the defined steps documented on the website [137]. In addition, the website elaborates a more thorough explanation of the mentioned notations alongside illustrative examples, as part of the entire GitHub project.

At different runs, the tool may return different solutions for the same scenario, since there may be multiple possibilities to verify the Boolean formulas from transitions of the SpecOPN. Therefore, for each experiment, a given number of simulations are performed to assess the quality of the results. The metrics chosen for the analysis of numerical results are the following.

- **(a) Model size**, as the sum of places and transitions, respectively, of all the representations assigned to the robotic team and the given specification.
- Average **(b) run time** to return a solution, based on all the simulations as part of one experiment, such that a given LTL mission is ensured by the robotic team. This metric excludes the time needed to build the model.
- Shortest **(c) trajectory length** for the whole robotic team obtained over all simulations performed within the same experiment. The trajectory length is expressed as the total number of fired transitions in the RobotOPN models.

Easy to follow example

This case study provides an altogether view of the defined formalism, considering the problem formulation from Chapter6.2. The planning strategy for a team of three robots evolving in a known workspace is visualized in Figure 6.5 for which a global LTL specification is given. The mission $\varphi = \Diamond b_1 \wedge \Diamond b_2 \wedge \Diamond b_3 \wedge (\neg b_1 \mathscr{U} b_3)$ implies the visit of regions of interest y_1, y_2, y_3, but requires that region y_3 be visited before y_1. The spatial constraints for agents impose an upper bound of two, i.e., $\mu_{cap}[c_5] = 2$, meaning that no more than two robots can be present at the same time in the cell c_5 modeled by $p_5^{o^1}$, $p_5^{o^2}$ and $p_5^{o^3}$, all labeled by y_1.

Figure 6.6 illustrates object nets. The left side (Figure 6.6-(a)), shows two different types of robots concerning their spatial capabilities: r_1 and r_2 are identical and are allowed

[2]Renew software tool: http://www.renew.de/

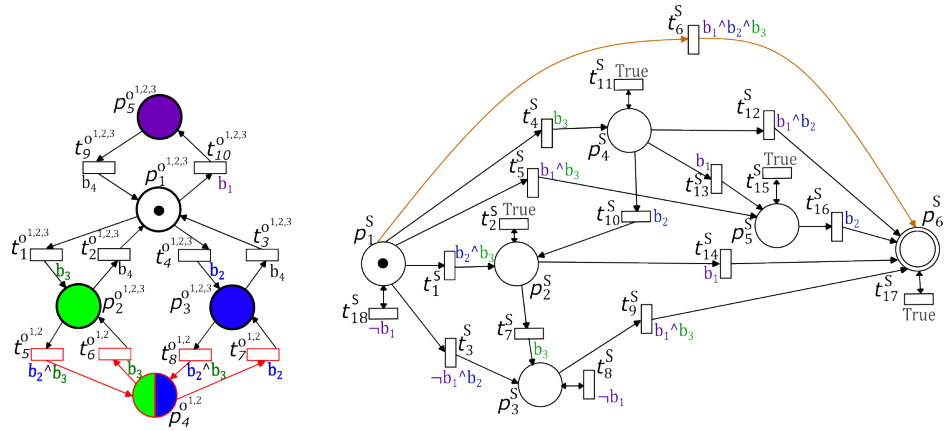


Fig. 6.6. (a) RobotOPN modeling three robots evolving in the environment in Figure 6.5. Two robots r_1 and r_2 can move freely in the workspace while r_3 is not allowed to enter the overlapped region between y_2 and y_3 (the model would be the same, but removing the red elements); (b) SpecOPN: the marked path corresponds to the shortest solution out of 100 simulations according to the trajectory length of the robotic-team

to move freely in the workspace. In contrast, r_3 is not allowed to enter the overlapped part of regions of interest y_2 and y_3 (illustrated by place $p_4^{o^{1,2}}$).

The right side (Figure 6.6-(b)) shows the SpecOPN model of the LTL mission φ, resulting from Algorithm 5. As mentioned previously, the results in terms of robots' trajectories are returned randomly by Renew. Therefore, we have conducted 100 simulations, with a mean execution time per simulation of $\mu = 14.25$ [ms].

Let us explain the orange run from Figure 6.6-(b). This run requires triggering a transition labeled $b_1 \wedge b_2 \wedge b_3$, which requires the simultaneous visit to regions y_1, y_2, and y_3. The mission is accomplished when place p_6^S in SpecOPN contains a token, guaranteed by the synchronization function GEF based on the movements of the robots in RobotOPNs.

Initially, the robots are in free space and cannot directly enter cell c_4 (intersection of green and blue regions modeled by places $p_4^{o^1}$ and $p_4^{o^2}$). Note that r_3 cannot enter this cell. Therefore, the only way to reach all three regions is for each robot to synchronously enter these regions in one step. Consequently, three robots must move, and transition t_3 in the system net must fire synchronously with t_6^S.

Furthermore, the three robots should move as follows: $t_1^{o^1}$ from o^1 labeled b_3, $t_4^{o^2}$ from o^2 labeled b_2, and $t_{10}^{o^3}$ from o^3 labeled b_1. Therefore, transition t_3 in the system net, which models the movement of three robots, will fire synchronously with $t_1^{o^1}$, $t_4^{o^2}$, and $t_{10}^{o^3}$ from the RobotOPNs, as well as with t_6^S. After these firings, a token will arrive in place p_6^S, fulfilling the LTL formula.

The motion planning obtained through the proposed algorithm does not guarantee an optimal robotic plan for the chosen metrics, influenced by the randomness of performing simulations in Renew.

6.5 Comparison with P/T Petri net

To provide a broader aspect of the benefits of the proposed (ii) *High-level robotic team Petri net* (HLrtPN), a comprehensive comparison example is provided considering the (i) P/T Petri net [151]. The example highlights the advantages of our proposed method for planning robotic trajectories, particularly in managing complexity. As the size of the robotic team increases, the P/T Petri net model scales exponentially, whereas the HLrtPN provides a more scalable solution. This comparison underscores the effectiveness of our approach in handling larger robotic systems.

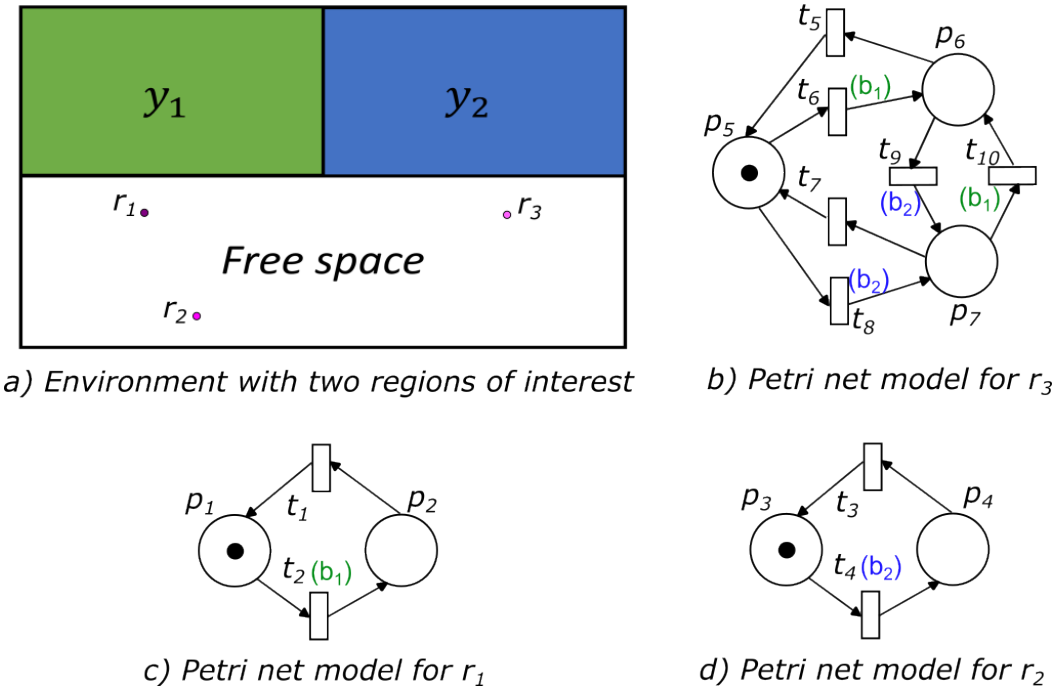


Fig. 6.7. Example of synchronous movements between the robots

Figure 6.7-a) shows a partitioned environment that includes two disjoint regions of interest y_1 (green) and y_2 (blue). In this workspace, a team of three robots is considered to evolve: one robot r_1 can reach only the region y_1, another robot r_2 is allowed to move only in the region y_2, while the third robot r_3 is allowed to move freely in the workspace. The Petri net representations for each robot r_1, r_2 respectively r_3 are captured in Figures 6.7-c), d), respectively b). The mathematical notations used for this example follow the notations defined in Chapter 2 i.e., set $\mathscr{Y}$ for the regions of interest and set $\mathscr{B}$ for the atomic propositions associated with the set $\mathscr{Y}$.

Each place represents a cell from the partitioned environment. Places p_1, p_3, and p_5 are associated with the free space, places p_2, p_6 are associated with region y_1, and places p_4, p_7 are associated with region y_2. The presence of a token in a place represents the position of the robot in the environment. Initially, the robots are located in free space.

The transitions that facilitate the entering of a robot in a region of interest have assigned a label equal to the value of the atomic proposition associated with the respective region of interest, e.g., b_1 for y_1, respectively b_2 for y_2. For example, the movement of robot r_1 from the free space in the region y_1 is portrayed by triggering the transition t_2 which evaluates the atomic proposition b_1 as *True* when consuming the token from place p_1 to p_2 (Figure 6.7 -c)). Contrary, when transition t_1 is fired (the robot entering the free space), the value of b_1 is unknown. The negation of any atomic proposition is a global variable, depending on the state of all robots, e.g., robot r_1 exits region y_1 while r_3 is present in the region y_1.

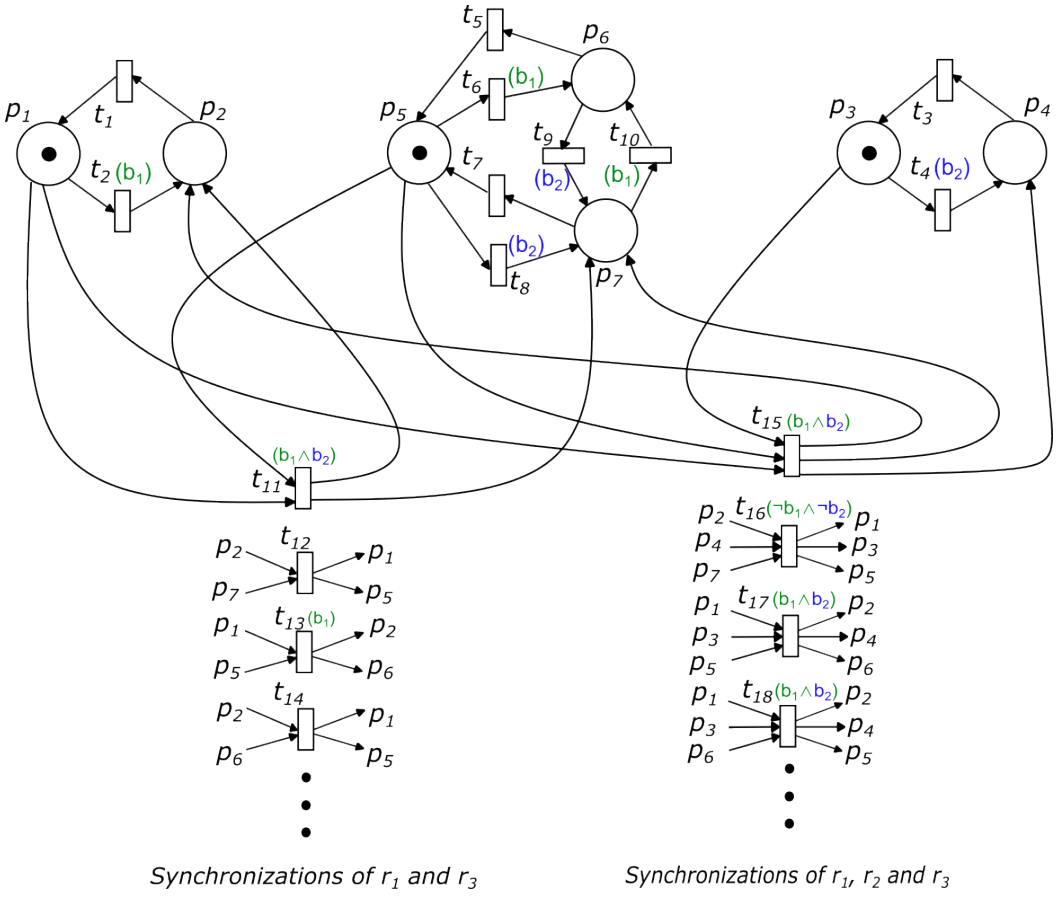


Fig. 6.8. Example of synchronous movements between the robots

Approach (i): P/T model. The P/T-net (also known as "Place/Transition Petri net") is a model where the tokens cannot be differentiated between them, as stated in[151]. Based on the studies from literature, the Petri net model is usually considered for motion planning solutions of homogeneous robotic teams, that should ensure a global mission. This aspect is given by the fact that the robots are represented by tokens, as in [107].

The global view of the robotic team can be visualized by adding several transitions in the P/T model, capturing the synchronous movements of the robot. For a clearer visualization, Figure 6.8 provides a partial view of the totality of transitions that should be added,

considering all the synchronizations of the robots. For example, the firing of transition t_{11} triggers a synchronization between robots r_1 and r_3, by evaluating the *True* value of $b_1 \wedge b_2$ when both robots enter regions y_1, respectively y_2. On the other hand, it can be observed that no label was assigned to transition t_{12} since this models the exit of robots r_1, r_2 from regions y_1, y_2 and enter the free space. The synchronization between r_1 and r_3 but not r_2 cannot yield any global information about the atomic propositions, associated with the movement of all robots. The number of synchronizations between r_1 and r_3 is 12, given by 2 transitions in the model of r_1 connected with the 6 transitions in the model of r_3. Similarly, there are needed 12 transitions for counting the synchronizations between r_2 and r_3.

The new transitions illustrated on the right side of Figure 6.8 show synchronizations of all robots in the team. In this scenario, transition t_{16} yields global information about the state of the robotic team, since it considers their synchronization. By firing t_{16}, both b_1 and b_2 associated with regions y_1 and y_2 are evaluated as *False*. The total number of synchronizations is given by the product of all number of transitions for each model: 2 (transitions for r_1) $\times$ 2 (transitions for r_2) $\times$ 6 (transitions for r_3).

Observe that in the provided example, where only three robots are in the team, the number of transitions that should be added is equal to 52, composed of 4 transitions for synchronizations between r_1, r_2; 12 for synchronizations between r_1, r_3; 12 for synchronizations between r_2, r_3; and 24 for synchronizations between r_1, r_2, r_3. Moreover, we have analyzed the synchronization between the robots, without considering another Petri net model for the given mission. In such a case, a large number of arcs should be added to the overall model, such that the motion of the robots updates the state of the mission.

Approach (ii): Nets-within-Nets paradigm. The main idea of this formalism is to facilitate a hierarchical structure of Petri nets: a system net provides the global information about the system, while the object nets represent the local state, particularly of the robots. Each robot is modeled as a Petri net, similar to the previous approach the difference being that these Petri nets are denoted *Robotic object nets* (RobotOPN), with o^1, o^2, respectively o^3 denoting robots r_1, r_2, respectively r_3. These nets represent the tokens for an upper-level Petri net denoted *System net*, particularly in the place *Rb*, visualized in Figure 6.9. The notations of places and transitions reflect the mathematical notations from our proposed method, $p_1^{o^1}, p_1^{o^2}$, and $p_1^{o^3}$ modeling the free space of robots r_1, r_2 and r_3. Observe that the *System net* contains a second place *Mb* carrying information for the specification given to the robotic team. In this example, we focus on the synchronization between robots, without analyzing the complex scenario of the robots ensuring a given mission.

In our framework, the free space is considered as a region of interest associated with an atomic proposition, specifically b_3 for this example. Therefore, the transitions triggering the entering in the free space, are labeled with b_3. Note that this labeling function is considered for all transitions that require synchronization, as part of the Nets-within-Nets paradigm [58].

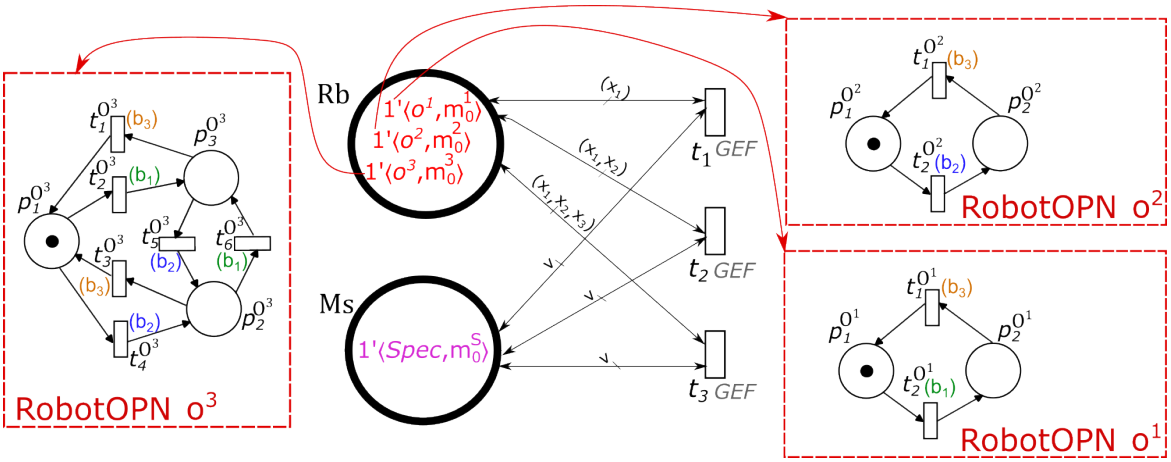


Fig. 6.9. Nets-within-Nets paradigm: Example of synchronization of the robots modeled as RobotOPN and coordinated by the system net and synchronization function GEF.

The synchronization function requires the values of the atomic proposition based on the enabled transitions in each local model. The system net manipulates its object nets by firing one of the transitions t_1, t_2, respectively t_3 associated with the synchronizations of one, two, respectively three robots through variables x_1, x_2 and x_3. Let us consider the following use-case: r_1 is in region y_1, r_2 in region y_2, while r_3 does not move from the free space. Currently, the enabled transitions are $t_1^{o^1}, t_1^{o^2}, t_2^{o^3}, t_4^{o^3}$. Since the Nets-within-Nets paradigm supports object-oriented operations, the synchronization function GEF acts as a guard by analyzing the state of the robotic team with respect to the specification net, before firing any transition. For example, if synchronization between r_1 and r_2 is required, while robot r_3 remains in the free space, then the function GEF provides information about the global state of the robotic team when transitions $t_1^{o^1}, t_1^{o^2}$ are fired. This leads to the update in the atomic propositions value: b_3 becomes *True*, while b_1, b_2 becomes *False*.

Discussion. The method (i) based on P/T net has been studied for system modeling, e.g., Silva, Manuel. "Introducing Petri nets." Practice of Petri Nets in Manufacturing (1993): 1-62; Murata, Tadao. "Petri nets: Properties, analysis and applications." Proceedings of the IEEE 77.4 (1989): 541-580; Peterson, James Lyle. "Petri net theory and the modeling of systems". Prentice Hall PTR, 1981. One main drawback is the complexity of the model obtained for complex systems being challenging to handle any modification and update when necessary with every modification in the robotic team.

As a general rule, the number of total transitions required to be added (provided that the global state of the robotic team) increases exponentially. This increase leads to a hardly malleable model. Particularly, let us denote the number of transitions for each robot $r_i, i = \overline{1,n}$ with $N_1, N_2, \ldots N_n$, where N_i is the number of transitions associated with robot r_i. The total number of new transitions N_T required for all synchronizations between robots is given by the following equation:

$$N_T = \left(\sum_{i=1}^{n}\sum_{j=i+1}^{n} N_i * N_j\right) + \left(\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n} N_i * N_j * N_k\right) + \cdots + (N_1 * \cdots * N_n) \quad (6.1)$$

On the other hand, the second method (ii) is based on Nets-within-Nets. This formalism facilitates easier modeling and handling of the entire representation, through its hierarchical structure of nets, while the object-oriented operations support the insertion of user-defined methods assigned to the system net and object nets.

We conclude that modeling a planning solution solely on P/T Petri net formalism is proving to be difficult to handle while tracking the correctness of arcs connecting each place such that the robotic team ensures the given mission. On the other hand, the proposed method under the Nets-within-Nets paradigm, facilitates the modeling of local states and global states, through its hierarchical structure. The local states are represented by a set of *Object nets*, modeling the motion of robots and the specification. The *System net* provides the global state of the robotic team concerning their mission, having as tokens the object nets. In addition, the object-oriented operations that can be handled by the Nets-within-Nets formalism, establish the synchronization between the object nets.

Chapter 7

Developed software routines and applications

This chapter presents several simulations and applications that evaluate the results derived from the proposed methods throughout the thesis under the Petri net formalism. The main idea is to show the versatility of the high-level frameworks proposed for this thesis, by analyzing the planning strategies addressed to different robotic teams: based on identical UAVs, heterogeneous mobile robots, and a team of cobots.

Firstly, the implementation section describes the steps throughout the deployment in MATLAB and Renew software tools, considering three methods previously detailed throughout the thesis. Specifically, the MATLAB implementation aims to include in the open-source toolbox denoted RMTool the following methods based on the Discrete Event Systems formalism: the rerouting planning algorithm from Chapter 4.3 and the *Composed Petri net* framework from Chapter 4.2. Moreover, the Renew tool allowed for evaluating the soundness of the framework *High-Level robot team Petri net* under the Nets-within-Nets paradigm from Chapter 6.1.

Secondly, two simulation scenarios with real applicability are considered for the result analysis, introducing planning strategies for a team of identical UAVs whitening the roof of greenhouses, respectively considering both homogeneous and heterogeneous robotic systems to plan collision-free trajectories and allocating tasks envisioning a futuristic situation in the healthcare domain. Lastly, this chapter concludes with an experimental validation of the framework *Composed Time Petri net* model in the industrial domain, particularly for a manipulating application using two cobots ensuring mission with space and time constraints under the MITL formalism.

7.1 Deployment

One contribution of this thesis is represented by the implementation process since several methods are uploaded online to serve as open sources for researchers. By having access to these methods, the previously described methods are a stepping stone for further improvements in the robotic path planning field. Thus, this section describes the implementation process, that has been considered throughout this thesis, aiming for a conceptual understanding of the software tools chosen for validating the formal methods presented in the previous chapters. Particularly, this section is divided into three phases, detailing the following: (a) MATLAB implementation [152] with a focus on the *Composed Petri net* formalism (Chapter 4.2) and the path rerouting method (Chapter 4.3) which is available in the toolbox RMTool [2], (b) Renew implementation [153] taken into account for the *High-Level robot team Petri net* system under the Nets-within-Nets paradigm (Chapter 6.1)), a deployment which is described in [137], a website that includes also the GitHub functions used for validating the proposed framework, and (iii) Romeo implementation [134] weighted for evaluating the Composed Time Petri net model (Chapter 5), since this tool allows for on-the-fly model-checking suitable for Time Petri net representations.

7.1.1 MATLAB implementation

The implementation was carried out in the toolbox RMTool in MATLAB [2], which can be found on this link [154]. Particularly, this toolbox is open-source, has a user-friendly graphic interface, and facilitates an easier deployment and validation of the proposed methods under the Discrete Event System representations, such as Petri net or Transition Systems. The implementation is reproducible and adaptable to the user's needs. Therefore, one of the contributions of this thesis relies on the availability of the implementation of the mentioned methods.

The methods described in this thesis and deployed in MATLAB could be easily validated, through its simplicity of expressing optimization problems with the Optimization Toolbox [155]. Moreover, the comparison with other algorithms from the literature (as stated in Chapter 2.4, is simple to assess, since the evaluation of the proposed and other DES methods are integrated in the same workspace.

Figure 7.1 represents a top-level overview of the implementation in MATLAB: (a) the RMTool yields an easy-to-handle graphical interface that includes (b) multiple path planning methods further implemented by (c) a set of MATLAB scripts. Among the methods available currently in RMTool, this section highlights the contributions related to the deployment of two of the proposed methods explained in this thesis, notably (i) the path rerouting algorithm from Chapter 4.3 which expanded on a previous path planning solution included in the RMTool, and (ii) the *Composed Petri net* framework from Chapter 6.1 which was analyzed in comparison with several algorithms under Discrete Event System's representation, algorithms

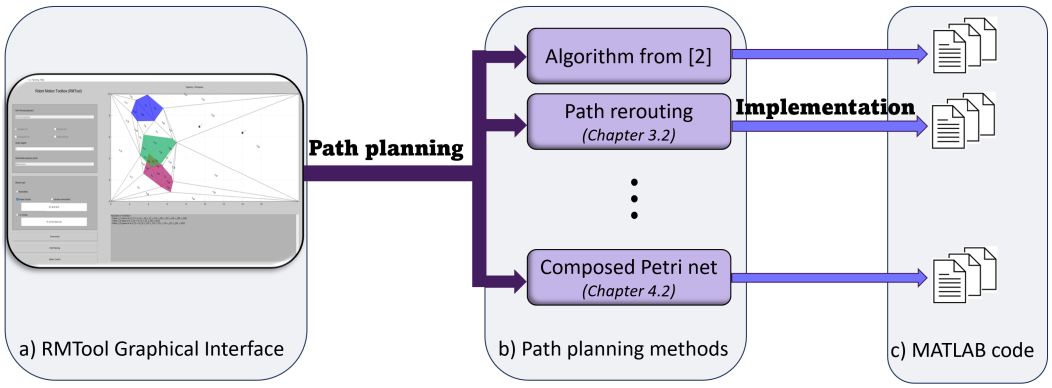


Fig. 7.1. Diagram of MATLAB deployment in RMTool

that are also included in RMTool. A short code description will be described for each of these two deployments, following the steps that should be considered for running the algorithms.

Figure 7.2 illustrates the graphical interface of the toolbox RMTool. On the right side (highlighted with light blue), it can be visualized the workspace including a set of regions of interest $\mathscr{Y}$ which is further partitioned into cells for an easier handling of the environment, and the positions of the robot. On the bottom side, a short mention of the cells modeling the regions of interest can be visualized. On the left side, different settings can be selected by the user, such as the type of cell decomposition technique, planning in a known or partially unknown environment, specifying the global mission of the robotic team described either as a Boolean formula (emphasized by the green color), either as Linear Temporal Logic specifications (as described in Chapter 2) and selecting the path planning strategy. The path planning algorithms can be chosen by pressing the button highlighted with purple, including methods published in [2, 4, 50] among others, such as two of the proposed algorithms from this thesis noted by (i) and (ii) in this section.

Moreover, on top of the graphical interface, a list of settings is accessible to the user. Among these, the user can add or remove robots once they were added initially in the workspace, to move their initial position and to modify the font size which enumerates the cells returned by the partitioning technique. In addition, the interface facilitates the choice for a particular optimization solver such as: *intlinprog* incorporated in the MATLAB tool [152], *GLPK* (GNU Linear Programming Kit) [156] or CPLEX [110]. Other inputs relevant to a path planning method, such as user-defined parameters, e.g., the number of intermediate markings for MILP 4.2, can be modified. For small or less complex problems, the *intlinprog* is efficient and convenient since it is integrated into MATLAB. For more advanced applications or large-scale MILP, CPLEX is usually the best choice, as observed throughout simulations when evaluating the proposed methods. GLPK provides a cost-effective solution but is unsuitable for large problems, such as having many robots on the team. Throughout the result analysis of the methods proposed for this thesis, the selected solver was CPLEX 12.10 which is compatible with MATLAB 2019b.

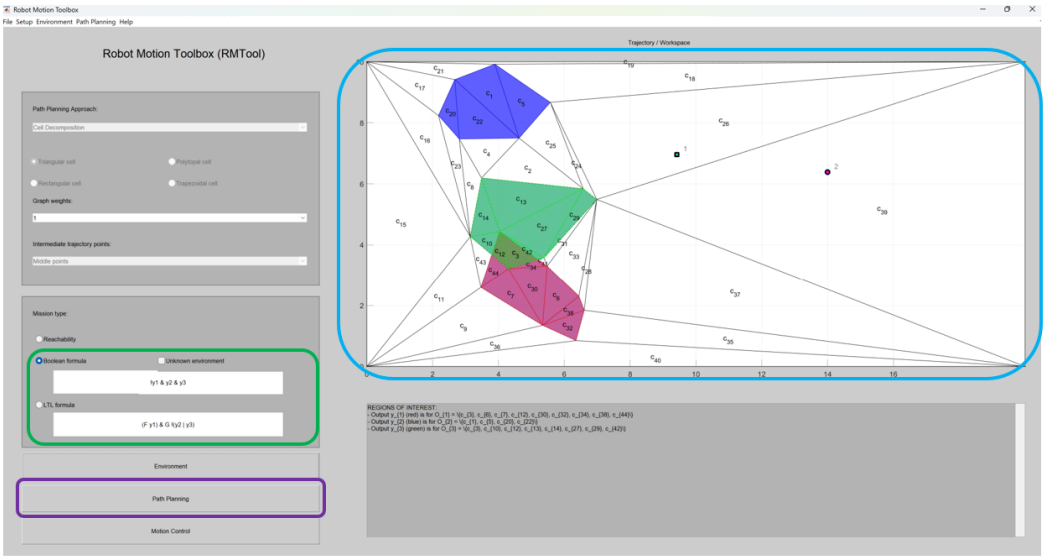


Fig. 7.2. Graphical interface of RMTool [2]

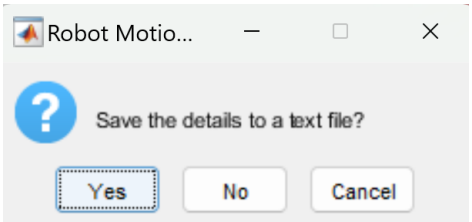


Fig. 7.3. Message addressed to the user regarding the saving of data based on the selected planning method

RMTool incorporates a feature that allows the user to save data about the simulation in .txt file, among which several pieces of information can be enumerated such as the run time for building and solving an optimization problem, the solver that the user selects, the size of the model (Petri net or Transition System) for which the planning method has been implemented on, the accepted run in the Büchi automaton that is followed in case of providing a global LTL mission, the trajectories of the robots ensuring the mission (LTL or Boolean) considering the cells that are being crossed through their paths. Figure 7.3 shows the message that the user receives by the interface once the path planning procedure is finished and the following text illustrates the data returned by the tool. This text considers a planning procedure based on the method from [2] for the Boolean mission $\neg b_1 \wedge b_2 \wedge b_3$ ensuring the visit of regions y_2, y_3 and the avoidance of region y_1 by a team of two robots.

```
Petri net system has 42 places and 122 transitions
Time spent for creating it: 0.0011535 secs
The MILP for intermediate state to satify the formula on
trajectory has 495 variables and 126 equality constraints
and 132 inequality constraints;
Time spent for creating the problem: 0.0014024 secs

The MILP solution is with GLPK

Time of solving the MILP for computing intermediate marking
to satisfy the formula on trajectory: 0.0061338 secs

Initial marking [ [21;27] ] = [1;1]
Boolean variables in solutions for the intermediate state are:
[0;0;0]

The MILP for the final state computation has 659 variables
and 168 equality constraints and 219 inequality constraints;
Time spent for creating the problem: 0.0036597 secs

The MILP solution is with GLPK

Time of solving the MILP for computing the final marking
to satisfy the formula on the final state: 0.0094235 secs

Intermediate marking [ [21;27] ] = [1;1]

============STEP 1 ============
```

```
Marking [ [3;23] ] = [1;1]

Sigma [ [54;61;69;75;78;80] ] = [1;1;1;1;1;1]
Boolean variables for the final state in solutions are:
[0;1;1]
Required number of synchronizations: 2

SOLUTION - runs of robots:

Robot 1: 21,19,3,3,3
Robot 2: 27,26,24,28,23
The number of steps for all robots is: 4
==========
The order of the robots is:  1  2
```

(i) Path rerouting implementation

Let us recall the path rerouting algorithm explained in Chapter 4.3, build upon the approach presented in [2], where a set of robotic paths is returned ensuring a Boolean mission specifying the final destination for the multi-robotic system while avoiding a set of obstacles along their trajectories. The proposed solution achieves parallel movements of the robots when the team should pass through a narrowed passage, maintaining the fulfillment of the Boolean mission, although the paths are rerouted to allow a parallel motion.

As mentioned before, the approach from [2] is implemented in the RMTool in the script denoted `rmt_path_planning_boolean_new`. Thus, by deploying the algorithm of the path rerouting into the same toolbox in MATLAB, the evaluation process ensures a controlled comparison between methods since the environment is consistent and can be reproduced.

Figure 7.4 shows the workflow diagram of the implementation code which provides an extended explanation of the Algorithm 8 by illustrating code fragments. The inputs (purple color) for this algorithm are represented by a Boolean mission for the robotic team and the environment designed by the user in RMTool, including the set of regions of interest that should be reached and/or avoided (based on the mission), the number of robots in the team and their location in the workspace, and select the type of cellular decomposition (as mentioned in Chapter 2.1). The tool also allows for a random positioning of the regions of interest and the initial position of the robots.

As it can be in the diagram (Figure 7.4), based on the delineated workspace, an RMPN representation (Definition 2.2.2) is built through the function that outputs the matrices *Pre* and *Post* based on the adjacency matrix resulted from the cell decomposition technique, specifically `rmt_construct_PN`. The pseudocode for this translation is explained in

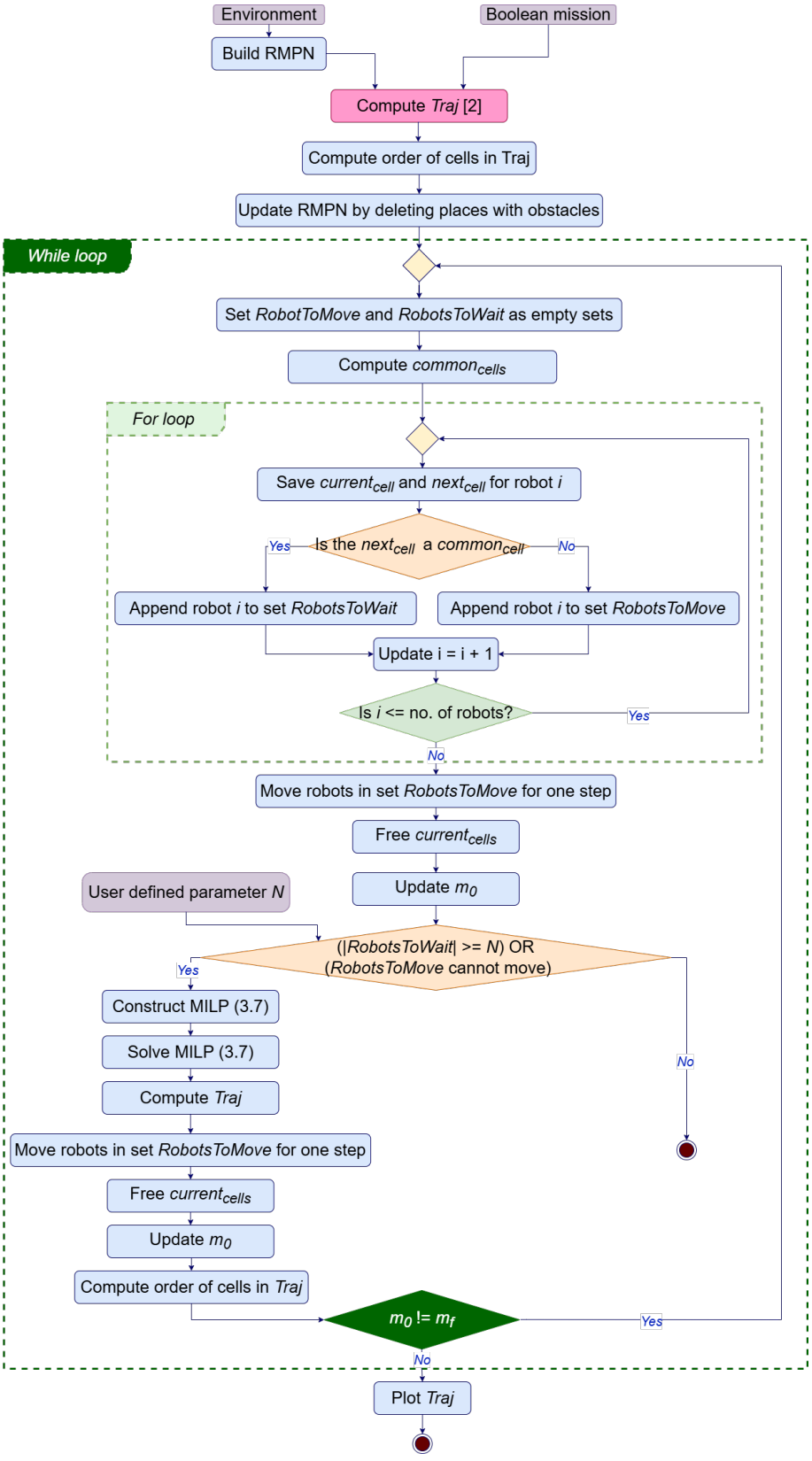


Fig. 7.4. Flow diagram for the method described in Chapter 4.3

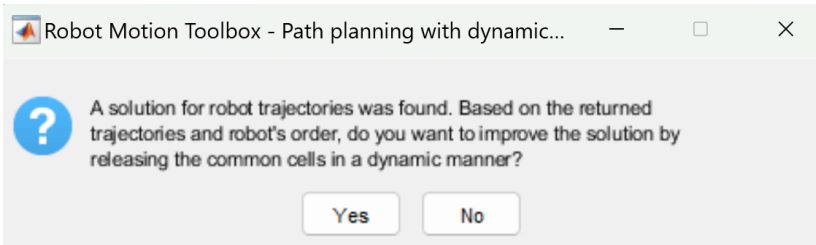


Fig. 7.5. Message to the user with respect to the rerouting procedure

detail in [1] (particularly, in Algorithm 4.2), where a place represents each cell, and the transitions are added based on the adjacency matrix.

The following step in the diagram is emphasized with the pink color, since the trajectories of the robots are computed based on the method proposed in [2]. The script in MATLAB contains four main functions: `rmt_construct_constraints_intermediate` which builds an optimization problem for the intermediate marking, meaning that the regions of interest along the trajectory are avoided as stated in the global Boolean mission, `rmt_construct_constraints_final` which builds the optimization problem for the final marking that ensures that the robots reach the regions of the interest, `rmt_path_planning_boolean_milp` which solves the previous optimization problems, and the last function allows the user to interpret the solution into robot trajectories `rmt_path_planning_boolean_trajectories`

To visualize the trajectories computed by the algorithm [2], it is recommended to press the path planning button (highlighted with purple in Figure 7.2). Immediately afterward, the user has the possibility of selecting another planning strategy based on the method from Chapter 4.3, suitable for a dynamic planning route based on the previously returned trajectories. Specifically, Figure 7.5 shows the message that the user visualizes it since the method that reroutes the trajectories contributes to a parallel motion of the robots fitted in scenarios where the robots should pass through a common free area to reach the specified regions of interest.

All the actions encapsulated inside the workflow diagram, starting from the *Compute order of cells in Traj*, are implemented inside the customed MATLAB function `rmt_path_planning_dyn_release_resources`. The name of the function accounted for the scenario where the narrowed passage includes resources in the form of common free cells that the team of robots should share among them while computing collision-free trajectories. The $common_{cell}$ is represented by a vector `cell_idx` that stores all the common free cells that robots should pass through, while $current_{cell}$ and the $previous_{cell}$ are associated with variables sharing the same name: `current_cell, previous_cell` for the current robot i, as it can also be seen in the *for loop* in the diagram.

Therefore, the set $Traj$ contains the ordered cells, returned by the following function `rmt_find_order_trajectories` and is saved in the variable `order_rob_cell`.

This step is necessary to observe the order in which the robots cross the common free cells since the method from [2] moves the robots sequentially. The update of the RMPN model inhibits the input and output transitions from the places that represent the obstacles in the workspace. The set *RobotsToMove* is computed in MATLAB in the variable `new_Run_cells` which is a cell with a size equal to the number of robots that should move. On the other hand, the set *RobotsToWait* is given by the variable `count` which is further used in the last decision node (with yellow), when the number of these robots is greater or equal to a user-defined parameter N.

The movement of the robots is visualized by the variable `flag_release` which is used afterward to release the resources, thus freeing the current cells of the robots. Once the movement is accomplished, the initial marking m_0 is updated to be further used to reroute the trajectories when either the number of robots that cannot move is greater than the user's threshold or the robots that should move cannot proceed. The second scenario occurs when a rerouting solution assigns the regions of interest to the robots through a set of trajectories. However, since the robots are moving in parallel, a blocking throughout the trajectory might happen. A visualization for this scenario is explained and shown in Figure 4.12 (b).

If the rerouting procedure is triggered by the conditions of the last decision node, then the function `rmt_path_planning_pn_bool_new_traj` build and solves the MILP (4.8). The solution is afterward interpreted and saved into the new set $Traj$ by the function `rmt_path_planning_boolspec_dif_trajectories`. Based on these trajectories, the robots that were previously in the category *RobotsToWait* are now relocated into the set *RobotsToMove* in order to advance for one step in their paths. Once all the robots that didn't reach their final region of interest moved for one step, the marking is updated in m_0 and the order of the cells is evaluated. The entire procedure is iterated until the initial marking m_0 is equal to the final marking m_f. In other words, the robotic team ensured the Boolean mission by reaching all the specified regions of interest, through collision-free trajectories, while the robots moved in parallel.

The video from [130] contains a comparison between the method from [2] with the path rerouting procedure, as a result of the implemented procedure described in Chapter 4.3.

(ii) Implementation of the Composed Petri net model

The novel framework *Composed Petri net* (Chapter 4.2) joining the PN model representing the motion of the robots in the workspace with the Büchi automaton modeling a given LTL mission to the robotic system is part of the RMTool implementation as one of the contributions of this thesis. Moreover, the algorithm 7 providing the global view of the proposed solution the robotic motion plan was analyzed in comparison with other DES methods also present in the same toolbox.

In the following, a workflow diagram of the Algorithm 7 is explained in detail (Figure 7.6), by emphasizing the key aspects from the implementation code in MATLAB in the function `rmt_path_planning_ltl_pn_with_buchi`. The purple bounding boxes

represent the inputs of the algorithm associated with the environment (the set of regions of interest and the number of the robots in the robotic team), the given LTL mission, and the number of intermediate markings k, all of them being defined by the user in the RMTool.

The workspace is represented by a RMPN model (Definition 2.2.2) by using the MATLAB functions `rmt_quotient_T_new` which firstly make a quotient partition of the workspace by joining the cells sharing the same observation and afterward building the Quotient PN through `rmt_construct_PN`. On a similar note, the Büchi automaton associated with the LTL formula is returned by the tool *ltl2ba* [93], which can be installed and used in MATLAB through the costumed function `rmt_create_buchi`.

The *Composed PN* representation is built by the function `rmt_construct_PN_ltl` returning variables for the matrices *Pre* and *Post* together with the index of the transitions that are virtual as described in Chapter 4.2. The MILP 4.2, specifically the matrices A, b for the inequality, respectively Aeq, beq for the equality constraints and the cost function with the coefficients saved in the vector *cost*, is built by the customed MATLAB function `rmt_construct_constraints_ltl_wBuchi`.

The first computation of the solution of MILP 4.2 is needed for the *prefix*. The solution saved in the variable *xmin* is afterward resolved in MATLAB by the function `rmt_check_active_observations`. The aim is to obtain the last active observations resulting from the last motion of the robots coordinated with the *prefix* path from Büchi automaton. For example, if the input from the second to last state represented by the s_f requires the robots to reach the regions y_1 and y_2, and the region y_2 is reached by visiting the overlapping area between $y_2 \wedge y_3$, then the active observations are $b_1 \wedge b_2 \wedge b_3$.

The next step visualized by the decision node, is to verify if the last active observations are present in the self-loop of the final state s_f. If yes, then the *suffix* is represented by this final state, and there is no need to solve the MILP 4.2, resulting in a smaller run time for computing a robotic path in the *Composed Petri net* model. Otherwise, the suffix is returned by the same MILP 4.2 (function `rmt_construct_constraints_ltl_wBuchi`) for which the initial marking m_0 is updated based on the motion produced by the *prefix*.

After the entire run is computed $Run = prefix, suffix \ldots$, then the solution based on the *Composed PN* requires to be projected in the RMPN model of the workspace, since the *Composed PN* representation includes a reduced model of the environment. This is achieved by function `rmt_construct_constraints_ltl_project_sol` implementing MILP 4.3. If the solution cannot be projected as shown in Appendix **??**, then this solution is saved in a vector (`bad_sol_pr` caused by the *prefix* and `bad_sol_suf` caused by the *suffix*).

The entire *for loop* is iterated for each final state until a solution is projected through MILP 4.3. In case no solution is acquired, then the number of intermediate markings k is increased by 1 up to its boundary given by the size of the *Composed PN* under the variable U. When an iteration occurs in the *while loop* and the set of final states is considered once more

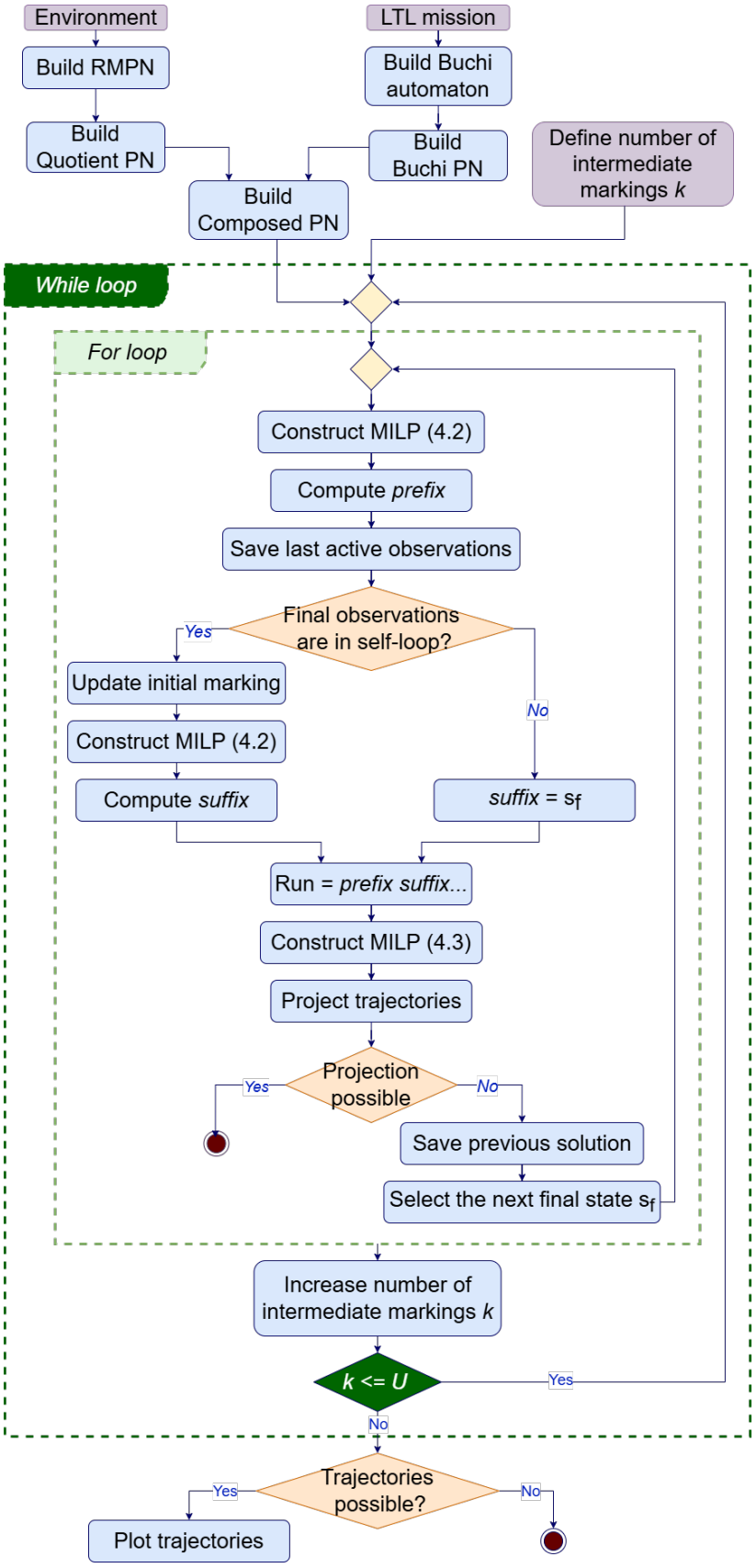


Fig. 7.6. Flow diagram for the method described in Chapter 4.2

for the planning procedure, then the MILP 4.2 ensures a new solution for *prefix*, respectively *suffix* which is different than the previous *bad solution* which could not be projected into the full RMPN of the workspace. Therefore, this action increases the expectations for the solution returned for a reduced model to be suitable for the extended one.

The last decision node verifies if a solution is possible using the *Composed PN* concerning the trajectories of a robotic system. If so, the trajectories are decoded through the costumed function `rmt_path_planning_ltl_with_buchi_trajectories` and are afterward plotted in the graphical interface. If a solution is not returned, then the user is informed about this fact in the .txt file, including the simulation information.

7.1.2 Renew implementation

In this subsection, a short presentation of the Renew tool shall be described, focusing on the key aspects that allow the user to model nets under the Nets-within-Nets formalism. Afterward, the open source implementation that is available on GitHub is outlined, following the Renew implementation process accompanied by illustrative examples considering the scenario from Chapter 6.1 for an easier and comprehensive understanding of this deployment. Finally, notes about running simulations shall be detailed such that any researcher to have a similar starting point for further improvements of the proposed novel model.

Renew (Reference Net Workshop) is a software tool Java-based simulator that excels in modeling and simulating Petri nets, particularly with its support for the Nets-within-Nets paradigm. This paradigm allows for hierarchical modeling, where individual nets can be encapsulated within larger nets, enabling modular design and complex system representation. The tool integrates the object-oriented properties further enhancing its flexibility. As a result, the users are allowed to define classes and objects that can be manipulated within the Nets-within-Nets paradigm [136].

Therefore, the implementation of the planning method using the *High-Level robotic team Petri net* model under the Nets-within-Nets (as presented in Chapter 6.1) is based on Renew. Particularly, the version of Renew is 4.1 due to the feature of compiling and simulating the nets using synchronous channels. [1]

As previously stated in Chapter 6.4, several details about the implementation are described in [137], which also contains references to two examples that are open access on GitHub: a simple scenario considering three robots and an easy to follow LTL mission [157] that is explained in Chapter 6.4 and a more complex scenario [158] considering a team up to 10 robots, which is further explained in Chapter 7.2.2. The following explanations accompanied by examples of modeling the nets in Renew are based on the simple scenario, to enhance the reader's understanding.

[1] The documentation of Renew includes the upgrades from one version to a newer one, as described here: https://www2.informatik.uni-hamburg.de/TGI/renew/4.1/renew4.1.pdf.

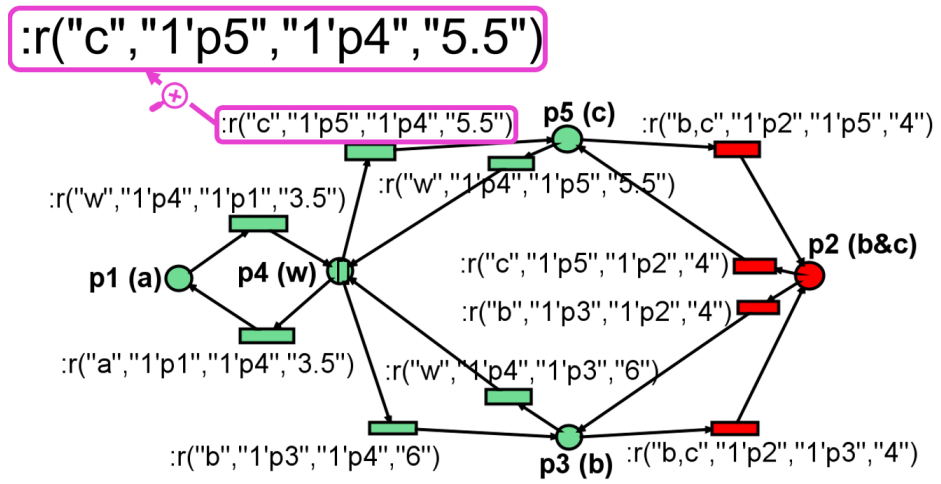


Fig. 7.7. Examples of the RobotOPN models in Renew: Robots r_1 and r_2 are free to move throughout the workspace. Robot r_3 is prohibited from entering the overlapping area between y_2 and y_3 (excluding the red places and transitions).

The GitHub projects include multiple types of files: the extension *.rnw* is associated with the Renew file for the designed nets (modeling the *RobotOPN*, *SpecOPN* and the system net, the extensions *.hoa* and *.pnml* are necessary when a net should represent the LTL mission associated with the *SpecOPN*, and the extension *.java* related to the Java scripts in which the Algorithm 10 is deployed. Particularly, there are two main Java scripts: `Eval` implementing the synchronization function between the system net and the object nets (visualized as tokens inside the system net), denoted *Global Enabling Function* (*GEF*), and `Perf_eval` computing the information about the solutions obtained after simulating the experiments in *.txt* file including the minimum and maximum steps required by the robotic team to ensure the given mission, the minimum, and maximum robotic moves by the entire team following an accepted run in the *SpecOPN*.

Let us recall the formal notation of the sets of regions of interest $\mathscr{Y} = \{y_1, y_2, \dots y_{|Y|}\}$, respectively of atomic propositions $\mathscr{B} = \{b_1, b_2, \dots b_{|B|}\}$. For a simpler visualization in the Renew simulator, the notations are redefined to eliminate the subscripts, e.g., the formal notation of atomic propositions $\mathscr{B}$ for the set of regions $\mathscr{Y} = \{y_1, y_2, y_3, y_4\}$ (Figure 6.5) is replaced here by set $\{a, b, c, w\}$, in exactly this order, with w assigned to the free space y_4. Additionally, the symbols $\neg$ and $\wedge$ are replaced in Renew with the syntax "!" and ",", respectively. The *True* value returned by the associated Büchi automaton of the co-safe LTL formula is represented in the tool by "1".

Firstly, Figure 7.7 portrays the design of a *RobotOPN* representation, considering the environment described from Figure 6.5. Specifically, the workspace contains a set of 3 regions of interest, two of them being overlapped $y_2 \wedge y_3$. In this workspace, there are two types of robots: r_1, r_2 moving freely and r_3 which is not allowed to enter the overlapped region. Thus, the place p_2 modeling the overlaid area together with the input and output

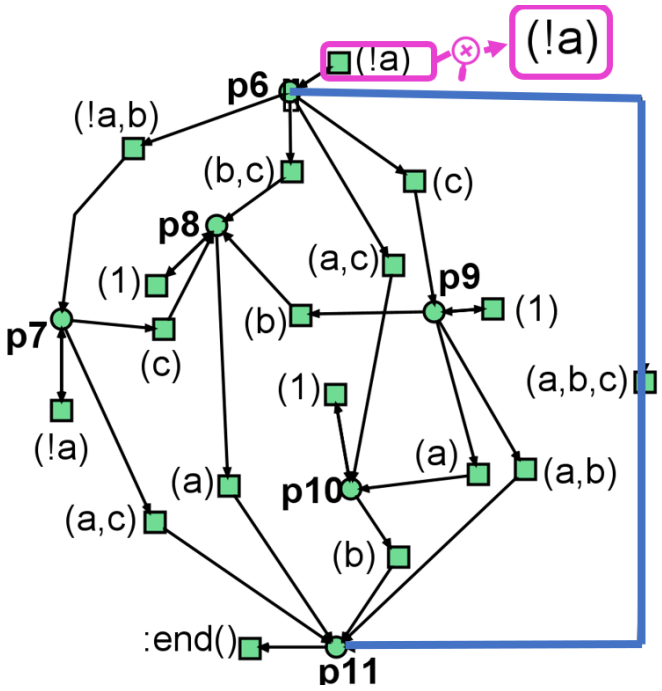


Fig. 7.8. Renew SpecOPN model for the LTL formula $\varphi = \Diamond b_1 \wedge \Diamond b_2 \wedge \Diamond b_3 \wedge (\neg b_1 \mathscr{U} b_3)$

transitions, are highlighted with the red color. In other words, the *RobotOPN* model for r_3 includes only the places and transitions colored with green, while r_1, r_2 is associated with a Petri net modeled by all the places and transitions illustrated in the figure.

Each transition is labeled, emphasized here by the magenta color. This following label $(c, 1'p5, 1'p4, ...)$ corresponds to the information required for the synchronization, used by the *GEF*: the robot occupies one unit in the region labeled with c (modeled by $p5$) while freeing its position from the free space w (modeled by $p4$). Thus, the atomic proposition for c is evaluated as $True$. The last parameter from the transition label represents additional information about the robot. In our example, the last data contains a number expressing the robot's time to move from $p4$ towards $p5$, e.g., 5.5 time units for the robotic movement, assuming that the robot has constant velocity.

Remark 7.1 In the implementation, there are two Renew models for these two types of robots. Since r_1 and r_2 have the same spacial constraints (moving freely in the workspace), a single RobotOPN model is necessary, including all 5 places, as visualized also in Figure 7.7. For r_3, the RobotOPN model in Renew includes only 4 places, excluding the reaching of the overlapped region modeled by $p2$.

Figure 7.8 illustrates the Renew modeling of the SpecOPN model. This representation is associated with the LTL mission $\varphi = \Diamond b_1 \wedge \Diamond b_2 \wedge \Diamond b_3 \wedge (\neg b_1 \mathscr{U} b_3)$. Similar to before, the magenta color emphasizes the label of transitions over the set of atomic propositions. Specifically, label $(!a)$ expresses the negation of the atomic proposition b_1 associated with region y_1. The blue color illustrates the run returned by Renew after the 100 simulations,

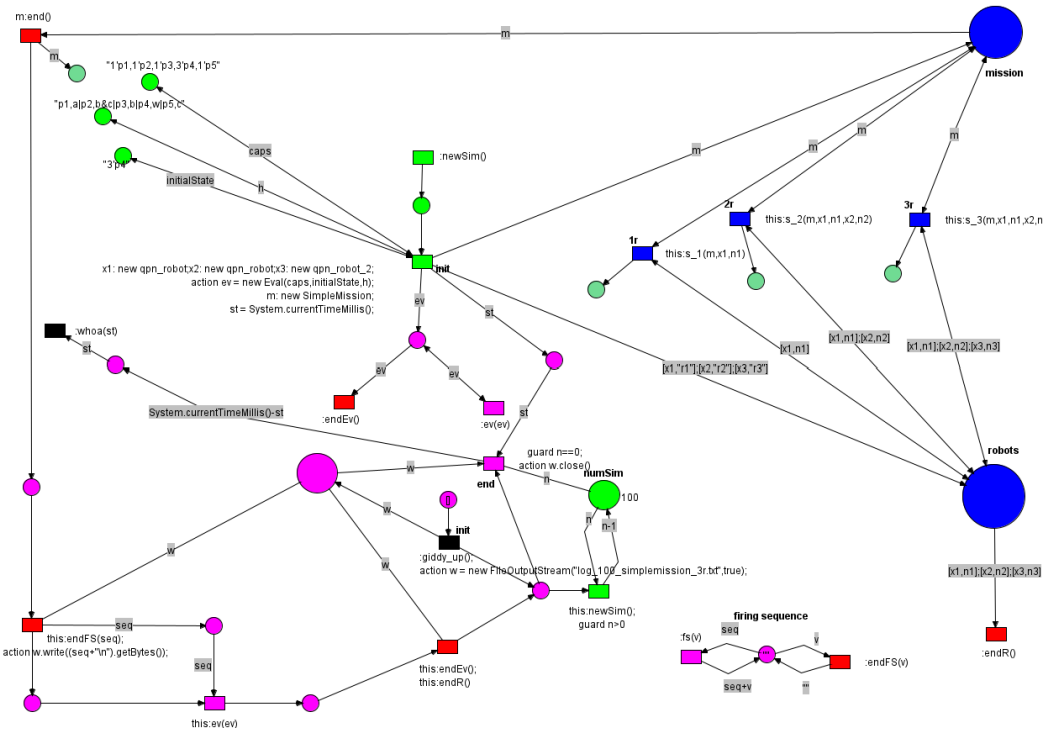


Fig. 7.9. Example of the High-Level robotic team Petri net model in Renew

representing the shortest path of the robotic team when the atomic propositions associated with region y_1, y_2, y_3 are reached in one step. With : *end*() is marked the place modeling the final state. In other words, a token in place $p11$ leads to the achievement of the mission.

Figure 7.9 portrays the main file representing the *High-Level robotic team Petri net* framework under the Nets-within-Nets paradigm. Specifically, the blue color illustrates the components of the system net: the place denoted *mission* includes as a token the *SpecOPN* model, while the second place *robots* includes three tokens referenced to the *RobotOPN* models. Furthermore, the blue transitions are added for each number of robots up to the maximum number of robots in the team, as detailed in Chapter 6 (Figure 6.4). The other colors used for this net are associated with the components necessary to initiate the simulation (color green), to finalize the simulation (color red), and to process the information concerning the shortest path of the robotic team and the run time for all the experiments (pink color). For example, on the top left side of the figure, three green places initiate the required inputs for the simulation such as the maximum allowed capacity for each place, e.g., 3 robots in the free space modeled by the atomic proposition *w*, the association between the places modeling the movement of the robots in *RobotOPNs* and their atomic propositions, e.g., p_4 modeling the free space *w*, and the initial position of the robots, all three robots being located in the free space.

To run the simulations, another net is modeled in Renew, necessary from a deployment point of view, denoted `execute_experiment`. The main idea of this net is to encapsu-

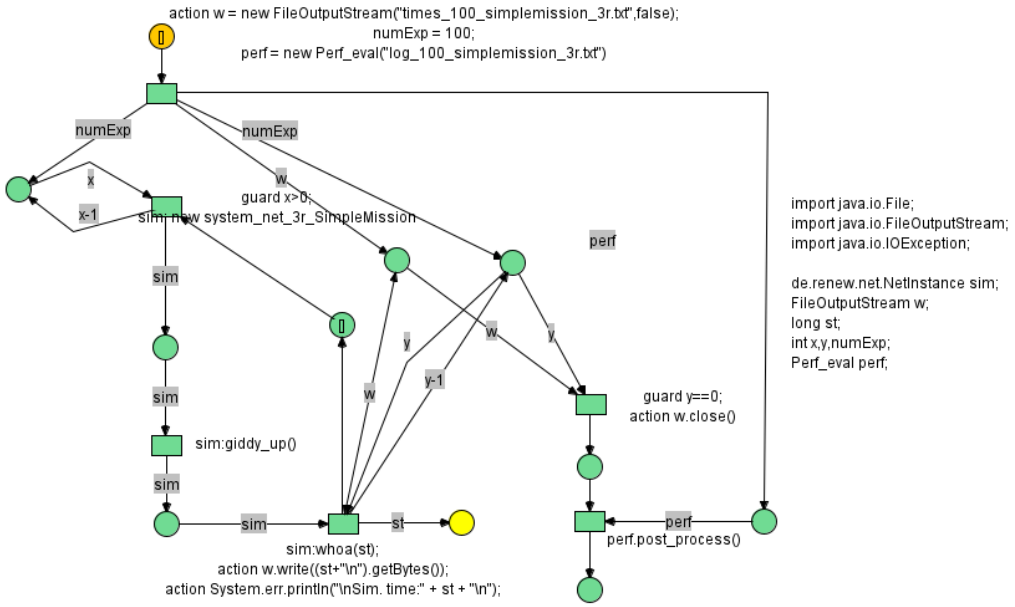


Fig. 7.10. Example of the `execute_experiment` file from Renew

late data about the simulation, considering the number of simulations considered for one experiment, the names of the *.txt* files that save information about the robotic path, and the total run time for one experiment in milliseconds. Figure 7.10 depicts this net. With orange color is expressed as the input place and with yellow is the end place. When the token reaches the yellow place, then the experiment finishes running and the data are saved regarding the result simulation.

The *SpecOPN* model can be represented directly by the user in the tool Renew. However, for complex missions, a manual design is difficult to build. Thus, the Renew tool encapsulates a characteristic that for any LTL mission to be expressed as a Petri net model in a file *.rnw* which can be further used for specific experiments that the researcher would like to analyze throughout the proposed framework *High-Level robotic team Petri net*. Thus, a list of steps is provided in the following:

- Translate the given LTL mission into a Büchi automaton through any model-checking tools [94, 93].
- Copy the detailed representation of the automaton in a file and save it with the extension *.never*.
- Convert the file to a new one with the extension *.hoa* by using the *autfilt* tool of the SPOT software [2]. Specifically, in the command line, the following command should run `autfilt NameOfTheFile.never > NameOfTheFile.hoa`

[2]The *autfilt* tool can be accessed on the following link https://spot.lre.epita.fr/autfilt.html.

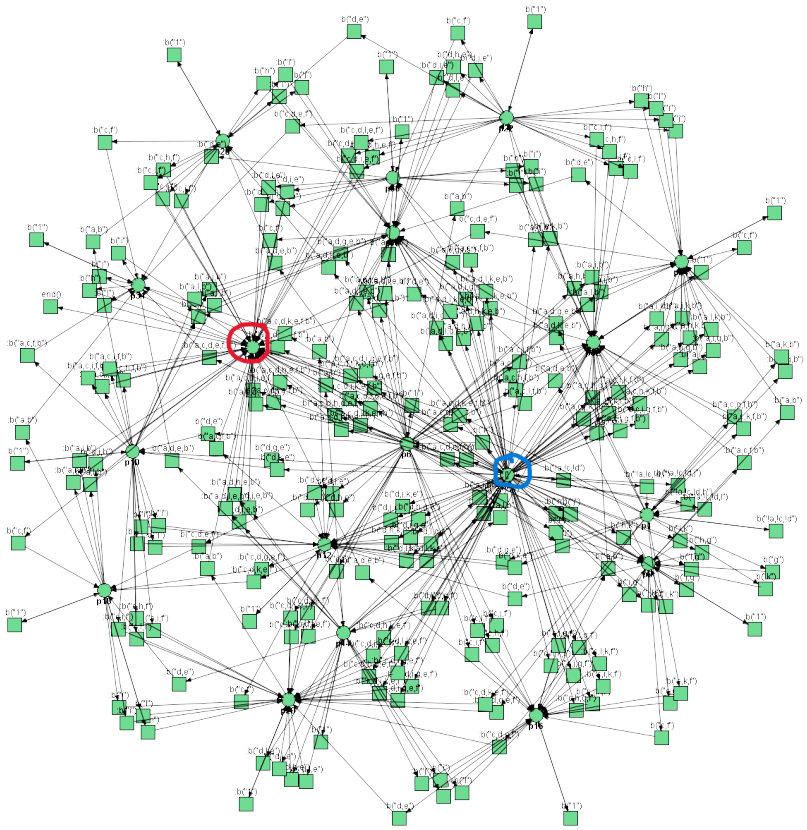


Fig. 7.11. Complex LTL mission φ modeled in Renew (blue - initial state, red - final state)

- Open a terminal window inside the folder *hoa2pnml* (from [157, 158]) and use the following command line `java -jar hoa2pnml.jar NameOfTheHoaFile`. The file should be added without the extension *.hoa*. Thus, the type of the file is converted to a *.pnml* which can be opened by the Renew tool and saved as *.rnw* to be further used as a *SpecOPN* model.

One example of a complex LTL mission is tackled in Chapter 7.2.2. Particularly, the LTL mission $\varphi = \Diamond(b_1 \wedge b_2) \wedge \Diamond(b_3 \wedge b_6) \wedge \Diamond(b_4 \wedge b_5) \wedge \neg(b_1 \vee b_3 \vee b_4) \mathscr{U} (b_7 \vee b_{11}) \wedge \Diamond(b_8 \vee b_9 \vee b_{10} \vee b_{12})$ express the visit of the first 6 regions of interest where two by two should be reached synchronously, followed by the visit of other regions up to region y_{12}. The Büchi automaton for this mission includes 18 states. However, since each transition is associated with a Boolean formula conveyed as a DNF (Disjunctive Normal Form), the entire model is returned by the conversion from the *.hoa* to *.rnw* contains 18 places and 264 transitions. Hence, this model cannot be easily designed manually in Renew. A visualization for this *SpecOPN* is shown in Figure 7.11.

The results of an experiment based on multiple simulations are saved in a *.txt* file, as presented below. For this example, the simulation results are analyzed and processed by the script `Perf_eval` mentioned previously, considering 1000 simulations. This file

contains the minimum and maximum time simulating the time for the robots to move from one region to another, as given in the transition labels as the last input. The maximum and the minimum steps refer to the number of movements made by the robots simultaneously, while the minimum and maximum robot moves count for the individual robotic movements of each robot. The example from the *.txt* file portrays a section of the entire result of the robotic team saved in the file, by considering the complex mission for a team of 4 robots.

```
min time:        27.5
max time:        1184.1000000000008
mean time:       307.54390470000004
minSteps:        5
maxSteps:        133
minRobotMoves:   15
maxRobotMoves:   405
```

7.2 Simulation results

Throughout this section, several case studies are introduced, considering various scenarios such as: (i) whitening the roof of greenhouses by a team of UAVs, problems addressed in the agriculture field; (ii) a futuristic hospital scenario where a team of heterogeneous mobile robots should synchronize with respect to a given mission. In both settings, the planning strategy relies on Petri net formalism, as previously detailed. These scenarios tackle essential issues in the field of robotic applications.

By 2050, according to the World Resources Institute, the demand for food will increase up to 25% compared with the present requirements [159]. Additionally, unpredictable weather and the effects of climate change will threaten food production and security. One possible solution to this problem is the use of greenhouses to intensify agriculture. Chapter7.2.1 proposes a planning strategy for a team of UAVs that should paint the roofs of greenhouses. The motion of the robots is given by a mathematical programming approach under Petri net formalism.

Mobile robots can enhance healthcare services in several ways, including optimizing operations such as delivering medications or cleaning rooms, minimizing human exposure to potentially contaminated spaces, lowering infection risks, and improving safety for both patients and healthcare staff [160]. Chapter7.2.2 includes a planning solution for a heterogeneous team that should respect synchronizations and sequencing for a given set of space restrictions, under LTL formalism. The robotic trajectories are returned by the Nets-within-Nets framework introduced in Chapter 6.

7.2.1 Whitening the roof of greenhouses by a team of UAVs

Motivation

The relevance of UAVs in the agriculture field is oriented towards (i) usage of spray systems [8], (ii) crop data acquisition and examination [9]. Besides these applications among others [10], several activities can be improved based on automated UAVs, having a beneficial impact in regard to human safety. One example is represented by the greenhouses in the southeast of Spain, in the province of Almeria.

The whitening of greenhouse's roof is crucial in controlling the amount of radiation that affects the crop, thus influencing the inside temperature in the greenhouse [161], right next to the natural ventilation [3]. The whitening process is made periodically and only when it is needed (the temperatures are high), while washing off the whitening when the natural ventilation is sufficient for the crop [162].

There are several problems with manually whitening the roof, the most important ones being related to the labor risks of the workers. For example, in [163], the authors report the labor accidents in the greenhouse-construction industry of SE Spain for the period 1999-2007 as 15133.7 accidents per 100000 workers per year. The most frequent type of accidents include cuts, punctures, contact with hard or rough material, overexertion, and falls from one level to another.

A set of risks can be enumerated as follows: (a) *falling risk*: dangerous action for a person since almost all covers (including the roofs) are made of plastic and can break very easily in case of a wrong step of the worker, (b) *adverse weather conditions*: since this process is done during the summertime around midday when the temperature is very high and the sun is shining. Moreover, in the (c) *case of wind*, when it changes direction or its force, particles of paint may land on the skin or eyes (if not protected) of the human operator. In addition to the human risks factor, one of the downsides of manually performing the whitening procedure lies in the *non-uniformity*, since the substance is not evenly spread as it would be through an automatic procedure.

One challenge is represented by the shape of the greenhouse's roof. For example, the usual type of greenhouse used in Almeria [3] is *"Raspa y Agamado"* (gable symmetrical modules roof) expressing 76.4% in 2013 out of the total types of greenhouses (this percentage increased from 62.5% in 2006). Other types are represented by Flat-Top 11.3%, Asymmetric 6.6%, and others 5.7% (this statistic was made in 2013). The ventilation of a greenhouse is directly influenced by its geometrical characteristics, hence the width is usually recommended to be below 30 m [164]. Considering that the size of greenhouses is not big, a particular challenge in Almeria is represented by the spread of greenhouses: 30.000 ha [165], part of which can be seen in Figure 7.12. Thus, the use of teams of UAVs is motivated by the scale and number of the greenhouses, in addition to the reasoning illustrated in the previous paragraph.

Problem formulation

Fig. 7.12. Greenhouses in Almeria [3]

Let us consider a 3D known environment including a greenhouse and several charging stations for a team of identical UAVs. Each drone has a finite flight autonomy which is being reset in the charging station. The given mission for the entire team specifies whitening the roof of the greenhouse, which can be interpreted as visiting and painting several regions of interest that include the surface of the roof.

In [166] it is proposed a solution for a similar application as the one tackled in this work, while the path planning problem is solved using a MILP formulation and transition system representation of the environment. This approach computes a large optimization problem due to sub-tour elimination constraints (a well-known issue in traveling salesman problems [167]). In addition, the number of unknown variables is dependent on both the size of the environment and the team of robots. The current work proposes to overcome the downsides expressed before.

The mission given to the team of UAVs ensures the painting of the entire roof of a greenhouse. This mission is embedded in the planning strategy, such that there is no need to express it formally using high-level formalism as LTL or MITL, as introduced in Chapter 2. As mentioned, the advantages of Petri net models rely on their scalability with respect to the number of robots, as it will be observed throughout the result analysis.

Example 7.2.1 *Figure 7.13 illustrates an example of the greenhouse with an even-span type of roof [168]. We assume that at each corner of the greenhouse, there exists a charging station together with a UAV. The drones are denoted as:* r_1 *- red UAV,* r_2 *- blue UAV,* r_3 *- green UAV,* r_4 *- magenta UAV.* ■

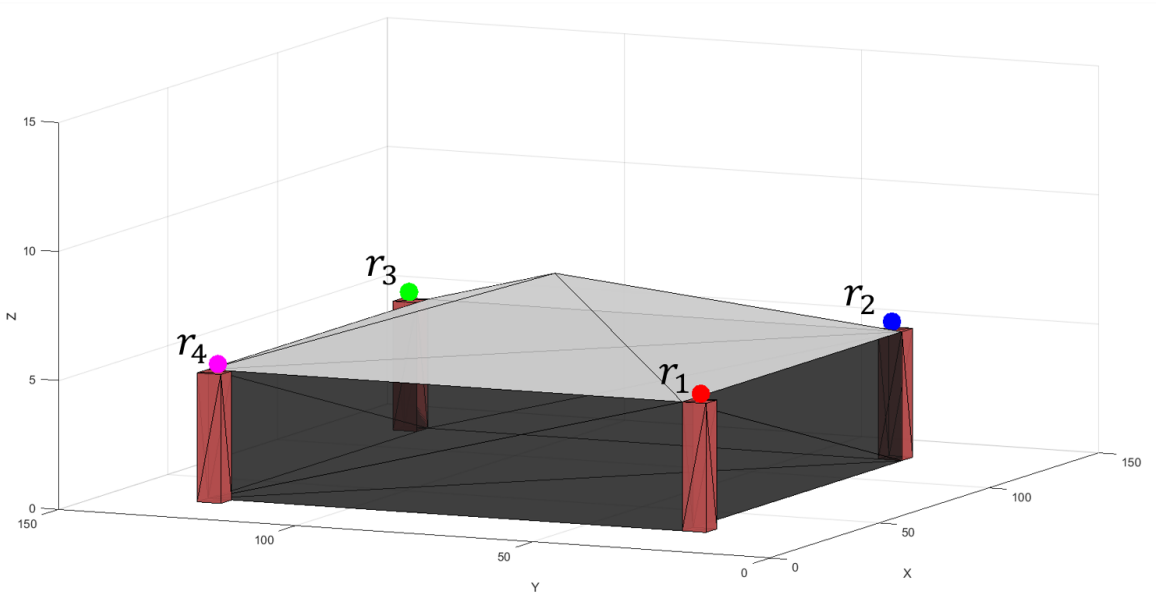


Fig. 7.13. Example of greenhouse environment with 4 UAVs

The space around the roof is divided into cells based on the 3D cuboid cell decomposition as presented in Chapter2.1. The idea is to divide the considered environment, e.g., the roof's surroundings, into cells that are further labeled as *Regions of interest* - the cells that should be visited and painted, intersecting the surface of the roof and *Free* - the cells which do not intersect the roof. For this work, let us denote with y_p a single region of interest including all the cells including the surface of the roof, and with y_f for the free space. The partitioning method is under the grid-based approach, where the cells have the same size. A Petri net model is built on the set of cells, with the following meaning: $h(p_i) = y_p, p_i \in P$ corresponding to the region of interest and $h(p_j) = y_f, p_j \in P, p_i \neq p_j$ corresponding to the region of interest. A place cannot be labeled both free and as a region of interest.

Example 7.2.2 *An example of a roof's space partitioned into cells is captured in Figure 7.14 for a precision* $\varepsilon = 4$*, resulting in a total set of* $4^3 = 64$ *cells. In addition, one cell was included for each initial position of UAVs to connect the roof space with the drones' charging stations. From the totality of computed cells, the ones which are fully intersected with the roof (the ones in which the UAVs cannot cross) are eliminated, resulting in a set of 60 viable cells.*

Figure 7.15 (X-Z view) illustrates the two different types of cells based on their labels: regions of interest represented by the cells intersecting the surface of the roof (green border) and free without any intersection (black border). For a clear visualization, only the cells relevant to one facet of the roof are captured here. ■

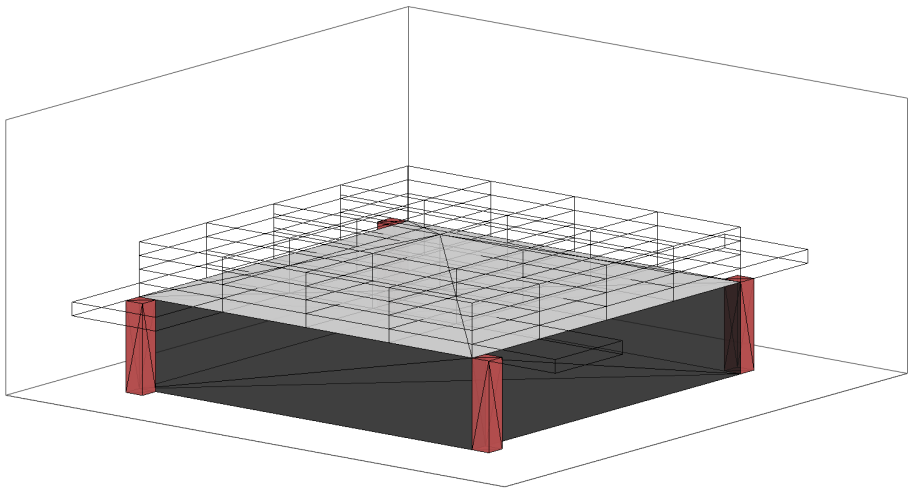


Fig. 7.14. Grid decomposition of greenhouse's roof, with precision $\varepsilon = 4$

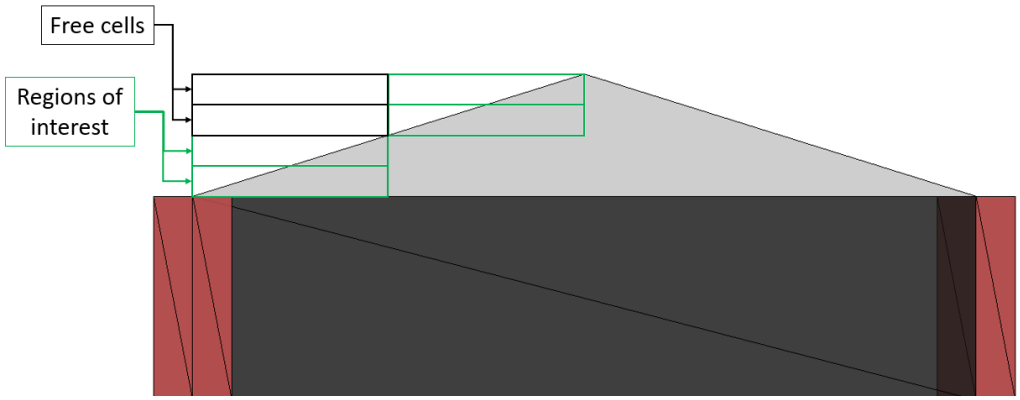


Fig. 7.15. Example of different types of cells: regions of interest and free

It is assumed that the drones fly in the space defined by the obtained viable cells (free and regions of interest), thus avoiding collision with the greenhouse.

Proposed solution

The approach consists of iteratively solving a Mixed Integer Linear Programming (MILP) (7.1) to compute the paths of all robots for a given tour. By a *tour* we understand the

paths of all robots starting from the charging station, paint some regions of interest, and return to the charging stations. The main idea of MILP (7.1) is to obtain a sequence σ such that as many cells (modeled by places) are visited, with $h(p_i) = y_p, p_i \in P$, subject to the energy constraint of the UAVs when they both move and paint the roof. The current work assumes the energy consumption constraints to be more rigorous than the paint capacity constraints. In addition, the MILP ensures that all UAVs included in the path planning problem reach a region of interest. The number of available UAVs for the MILP is denoted with $r_{Av} \leq |R|$. One refers to *available UAV* for MILP if the solution returns a trajectory towards regions of interest for the same UAV, i.e., an available UAV has enough energy to reach and paint at least one cell labeled as region of interest. The MILP is described as follows:

Variables:

- $m \in \mathbb{R}_{\geq 0}^{|P|}$ - marking of RMPN for the entire team of UAVs;
- $\sigma \in \mathbb{N}_{\geq 0}^{|T|}$ - firing count transition vector of RMPN.

The values of vector m are considered real for the MILP, to reduce the complexity for the solved problem when compared with an ILP formulation.

Objective:

$$\min \ 1^T \cdot \sigma \tag{7.1a}$$

Constraints:

$$m - m_0 - C \cdot \sigma = 0 \tag{7.1b}$$

$$E^T \cdot Post \cdot \sigma \leq r_{Av} \cdot \min_{i=1,r_{Av}} E_{c_i} \tag{7.1c}$$

$$L^T \cdot m = \min(r_{Av}, |P_p|) \tag{7.1d}$$

$$Post \cdot \sigma \leq 1 \tag{7.1e}$$

The MILP (7.1) minimizes the number of the fired transitions $1^T \cdot \sigma$. The constraints of this MILP are as follows,

- Constraint (7.1b) is the state equation (2.1).
- Constraint (7.1c): considers that the energy consumption for the entire team of UAVs along the paths is less or equal to an approximation of the entire available energy of the team. This entire available energy is approximated with the minimum energy of the available robots multiplied by the number of available robots r_{Av} (right-hand term). The left-hand term contains $E \in \mathbb{R}^{|P|\times 1}$ which is a vector containing the energy to cross free places (regions that shouldn't be painted), respective region of interest places (regions that should be painted).

- Constraint (7.1d) specifies that in the final marking m, all available robots should reach a region that should be painted. If the number of regions to paint is less than the number of available robots, then only a subset of robots are moving. Vector $L \in \{0,1\}^{|P|\times 1}$ is defined as $L[p_i] = 1$ if $h(p_i) = y_p, p_i \in P$, and $L[p_j] = 0$ otherwise.

- Constraint (7.1e) is responsible for the collision avoidance between UAVs, ensuring that only one UAV can be in one place throughout the trajectory, where 1 is a vector of $|P|$ elements having all elements equal to one.

The main advantage of using the PN model is emphasized through this MILP, e.g., when the team on UAVs decreases or increases, only the initial marking m_0 is modified, while the structure of the mapped environment remains the same. In addition, the size of MILP (7.1) remains constant, as the number of constraints is not dependent on the size of the UAV team, nor the size of the environment.

The previous MILP is incorporated into an overall algorithm, which is solved multiple times, based on the number of cells to paint. Algorithm 11 captures the path planning strategy for one tour. This procedure is centralized and iterated until all cells that should be painted are visited. Before executing the algorithm, the following values are initialized:

- $r_{Av} = |\mathscr{R}|$, i.e., initially, the number of available UAVs is equal to the number of UAVs. The loop in line 1 of the algorithm is iterated until the number of available robots is zero (all robots should be recharged), thus a tour is finished.

- $E_{c_i}, \forall i = 1,\ldots,\mathscr{R}$ is initialized with the maximum energy as all UAVs are charged fully with energy in the charging stations. As mentioned above, the paint capacity constraints are included in the energy consumption constraints. In this problem, the quantity of paint carried by the UAVs does not represent a restriction, since the flight autonomy of drones is considered more stringent than the carrying capacity.

- $Paths_i$ represents the path towards the regions of interest that should be visited and painted, computed by MILP for all UAVs $r_i \in \mathscr{R}$.

If MILP (7.1) has a feasible solution (line 3), an estimation of the current energy $\bar{E}_{c_i}$, for all available robots r_i, is calculated based on the returned solution of MILP (loop in lines 4-6). In line 7, if the estimated current energy of all UAVs is greater than a given threshold E_τ , then the paths returned by MILP for all available UAVs are computed. Moreover, the necessary parameters included in the optimization problem (lines 10 and 11) are updated. On the other hand, if at least one UAV doesn't have enough energy (compared with the threshold E_τ) or the solution is unfeasible, then a path towards a charging station is computed for the UAV with less available energy r_i (line 13). As a consequence of this action, the number of available UAVs for MILP is updated accordingly and the loop is iterated.

Result analysis

Algorithm 11: Path Planning for one tour

Input: RMPN $\mathscr{Q}$, regions to paint with $h(p_i) = y_p, p_i \in P$, number of available UAVs r_{Av}, current energy of each robot E_{c_i}

Output: *Paths*

```
while r_Av > 0 do
    Solve MILP (7.1);
    if solution is feasible then
        forall r_i do
            Compute the energy E_i of executing the path resulting from the MILP's solution;
            Ē_{c_i} = E_{c_i} − E_i;
    if solution is feasible AND min_{i=1,r_Av} Ē_{c_i} > E_τ then
        forall r_i do
            Append to Paths_i the path based on MILP;
            Update m_0, L, p_i with h(p_i) = y_p;
            E_{c_i} = Ē_{c_i};
    else
        For r_i with less E_{c_i} compute the path to a charging station and append it to Paths_i;
        Update m_0;
        r_Av = r_Av − 1;
```

The simulation results are obtained on a computer with i7 - 8th gen. CPU @ 2.20GHz and 8GB RAM after the algorithm was implemented in MATLAB. The selected solver for MILP (7.1) is CPLEX [110].

Example 7.2.3 *Let us recall the Example 7.2.1, illustrating a team of 4 UAVs which should whiten the roof of a greenhouse. For the considered simulation, the size of the greenhouse is* $100 \times 100 \times 5$ *meters [m], while the highest point of the roof is at 8 [m]. Each UAV is placed in a charging station at every corner of the greenhouse.*

The roof's space is modeled as a grid-based environment based on the 3D cell decomposition method. For precision $\varepsilon = 4$ *we obtained 60 viable cells (through which the UAVs can fly) in 0.24 seconds. In addition to these cells, 4 more cells are added, one for each initial place of UAVs. Therefore, the roof's space is captured in 60 cells, from which 32 represent regions to paint (ROIs). The whitening of the roof (interpreted as visiting all ROIs by the UAVs team), finishes in 3 tours, while the run time of one instance of MILP* (7.1) *with all robots available* $r_{Av} = |\mathscr{R}|$ *is 0.06 seconds.* ■

Table 5.1 capture scenarios where the size of the UAV team and the precision are modified. The following energy preconditions were assumed for these simulations: (a) for precision $\varepsilon = 4$ the energy to move between two adjacent places is 3%, while the energy for painting one region of interest represents 15% out of maximum (fully charged 100%) drone energy; (b) for precision $\varepsilon = 8$ the energy to move, respectively to paint consumes 1%, 5% and (c) for precision $\varepsilon = 16$ the energy to move is 0.5% and to paint is 2%. The energy consumption for moving/ painting one cell is reduced when the precision increases, due to smaller areas to paint.

Let us recall the fact that the previous work [166] considered a similar problem formulation. The differences between these two works consist in the chosen model of the environment: transition system in [166] and PN model in the current work. In addition, the previous work accounts for a single run of the MILP problem while including as input the number of tours, while the current work is based on an iterative approach to the MILP problem, the number of tours increasing every time when the first condition in Algorithm 11 is not verified. In [166] the results show an exponential increase in constructing and solving the MILP, based on the sub-tour elimination constraints assigned to each robot in the team and a large number of unknown variables. For example, in [166] the running time to compute MILP is 2359 seconds and to solve it is 152.36 seconds for a small environment with only 15 nodes in the graph. Simulations with more than 15 nodes were not computed because the computer ran out of memory, the trigger being represented by the large number of unknown variables in MILP. On the other hand, the current work provides a solution for a large environment containing 2980 nodes (places in the Petri net model), having the following performances: constructing the MILP in 20 seconds and solving the MILP in 13.26 seconds.

Moreover, the results in Table 7.1 show the same running time to solve the MILP when the precision ε is the same and the size of the team varies (the first two lines), as the optimization problem depends only on the size of the environment. Thus, the main advantage of the PN model is emphasized. To ease the visualization of the results, a video animation can be accessed at this link.

7.2.2 Assisting multi-agent robotic systems in healthcare field

Motivation

Let us consider a hospital procedure, e.g., MRI (Magnetic Resonance Imaging), suitable to scan images of the patient's body which are further used in diagnosing medical conditions or plan treatments. Due to the magnetic field generated by the machine, the computer used in the scanning process is in a different room. A radiographer usually operates the scanning process from another room. Depending on the body part that has to be monitored, the acquisition time varies, e.g., measuring the flow rates in vessels can take up to 30-40 minutes long [169]. Due to the time-consuming process of the scanning and monitoring,

Table. 7.1. Simulation results evaluation

Environment scenario	**Run time for cell decomposition [sec]**	**No. ROI**	**Run time for 1 instance of MILP** (7.1) **[sec]**	**No. tours**
r = 8 UAVs, precision $\varepsilon = 4$ PN model with $\|P\| = 60, \|T\| = 288$	0.24	32	0.06	2
r = 4 UAVs, precision $\varepsilon = 4$ PN model with $\|P\| = 60, \|T\| = 288$	0.24	32	0.06	3
r = 4 UAVs, precision $\varepsilon = 8$ PN model with $\|P\| = 404, \|T\| = 2328$	2.1	112	0.17	4
r = 4 UAVs, precision $\varepsilon = 16$ PN model with $\|P\| = 2980, \|T\| = 18856$	29.66	512	13.26	6

the researchers are inclined to automate it, e.g., the authors of [170] aim to close the gap between the current manual approach of ultrasound acquisition by using a robotic system. Since the tendency is to reach fully automated systems assisting in the medical field, many works provide different solutions approaching this aspect. One example is in[171], where various methods for the ultrasound procedure are structured based on a defined autonomy level.

Problem formulation

The need to automate this medical process among others, allows us to introduce a complex scenario suitable for motion planning of a robotic system with physical applicability in the real world. The main idea is to output robotic trajectories considering the proposed **High-Level robotic team Petri net** (HLrtPN) model under Nets-within-Nets paradigm, as presented in Chapter 6. The following scenario provides a wider perspective into the complexity in which the HLrtPN model enhances the scalability property of the Petri net formalism.

Let Figure 7.16 illustrate the layout of a hospital with three floors. The hospital includes a total number of rooms of 12, denoted by the set $\mathscr{Y} = \{y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}, y_{12}\}$, with examination rooms y_7, y_{11}, surgery rooms y_8, y_{12}, therapy rooms y_9, y_{10}, and MRI rooms y_1, y_3, y_4 which can be monitored only from rooms y_2, y_5, y_6.

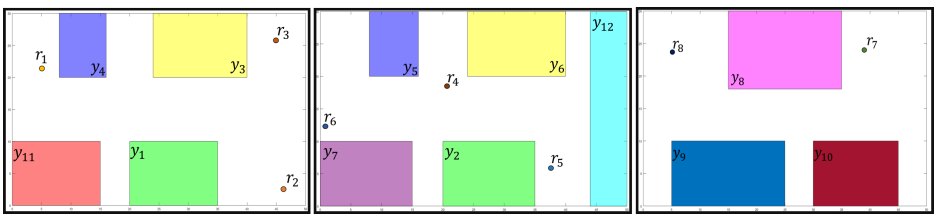


Fig. 7.16. Example of a hospital scenario with three layouts and 12 rooms for a multi-robot system.

Multiple rooms from the hospital are reached at one point in time. Firstly, the patients should be first examined in one of the examination rooms. If an MRI procedure is required, then the required rooms should be reached synchronously, as the patient is monitored simultaneously by a scanning robot. Any of the surgery and therapy rooms can be reached eventually, to be supplied and cleaned. The associated co-safe LTL mission is expressed in (7.2) correlated with the set of atomic propositions for each region from set $\mathscr{Y}$, i.e., b_1 associated with y_1.

$$\varphi = \diamondsuit(b_1 \wedge b_2) \wedge \diamondsuit(b_3 \wedge b_6) \wedge \diamondsuit(b_4 \wedge b_5) \wedge \neg(b_1 \vee b_3 \vee b_4)\mathscr{U}(b_7 \vee b_{11}) \wedge \diamondsuit(b_8 \vee b_9 \vee b_{10} \vee b_{12}) \tag{7.2}$$

The robotic system includes different types of robots, based on their spatial capabilities: r_p are robots carrying patients, r_m have scanning abilities only for the MRI procedure, r_{sc} are supplier and cleaning robots (supply with medicament and sterilize the rooms in which the patient should enter for medical operations) and r_a are assistant robots having a wide range of actions, realizing the tasks of r_m and r_{sc}. Table 7.2 illustrates the agents' capabilities w.r.t. the spatial constraints. For example, agents r_p can only enter rooms $y_1, y_3, y_4, y_7, y_{11}$ for MRI or leading the patients for examination, while agents r_m have access only in rooms y_2, y_5, y_6 to scan the patient during the MRI procedure.

Result analysis

The simulations are conducted on a computer with 12^{th} Gen. Intel®Core i7-12700x20 and Ubuntu 24.04LTS operating system, with 32Gb RAM, using Renew 4.1 [153] for the results based on the HLrtPN model under Nets-within-Nets paradigm, and MATLAB [152] for the rest of the methods as they are enumerated below.

Let us recall the methods previously described in Chapter2.4:

- (i) **FB [172]** - A sequential approach based on Petri net model that solves an MILP for robot trajectories by following an accepted run in the Büchi automaton. The iterative process yields sub-optimal solutions without ensuring collision-free paths.
- (ii) **TS [50]** - Each robot is represented by a Transition System model. A product model is composed out of these TS and the Büchi automaton of the LTL specification, such that a graph-search algorithm computes trajectories for the robotic team.

Heterogeneous robotic team

These two methods serves in the comparison analysis of the proposed HLrtPN model, since this scenario is complex and requires a deeper examination of the results. In addition, the comparative study includes also the proposed framework from Chapter 4.2, based on the **Composed Petri net** model. For simplicity, in the result table, let us denote this method with the abbreviation (iii) **CPN [35]**. As a reminder, this model introduces a parallel approach

where a reduced Petri net for robot motion and a Büchi automaton for the mission are integrated via an intermediate layer of atomic propositions. Two MILPs compute collision-free motion plans, though completeness is not guaranteed due to a projection step.

The result analysis is executed for teams of two to eight robots, shown in Table 7.3. The first columns of the table present the cardinality of each type of robot for every scenario. It is observed that the **(a) model size** directly influences the **(b) run time**. These simulations prove that the proposed framework satisfies the main objective in terms of a motion plan for a multi-agent system, considering offline planning. Thus, the **(c) trajectory length** could be shortened by introducing an optimality problem, the visualized result currently being computed through random solutions.

Remark 7.2 Generally, the proposed framework ensures solutions in which a subset of the robotic team synchronizes. This subset is a user-defined agent group bounded by the team cardinality. Particularly, the second case study for heterogeneous robotic teams generates solutions determined by a subset equal to the entire set of the robotic team (Table 7.3).

Floor	Rooms	Robots			
		r_p	r_m	r_{sc}	r_a
I	y_1	×			
II	y_2		×		×
I	y_3	×			
I	y_4	×			
II	y_5		×		×
II	y_6		×		×
II	y_7	×			
III	y_8			×	×
III	y_9			×	×
III	y_{10}			×	×
I	y_{11}	×			
II	y_{12}			×	×

Table. 7.2. Robots spatial capabilities considering the hospital's rooms.

Homogeneous robotic team

Let r_f be a full robot that is not restricted in movement and has access to all rooms. Explicitly, the robot can carry patients in the examination and MRI rooms, it has the necessary abilities to scan the patient for the MRI process, as well as being able to carry supplies and clean the therapy and surgery rooms. When the team includes only this type of robot, the team consists

No. of rob	Types of robots				No. of simulations	**(a) Model size**	**(b) Run time** [s]	**(c) Trajectory length**
	r_p	r_m	r_{sc}	r_a				
2	1			1	1000	$(\|P\|,\|T\|) = (42,294)$	0.39	27
3	1	1	1		1000	$(\|P\|,\|T\|) = (37,293)$	0.24	29
4	2	1	1		1000	$(\|P\|,\|T\|) = (44,305)$	1.1	25
5	2	2	1		1000	$(\|P\|,\|T\|) = (48,311)$	1.89	15
6	2	2	2		1000	$(\|P\|,\|T\|) = (54,321)$	10.4	14
7	2	2	2	1	250	$(\|P\|,\|T\|) = (69,337)$	107.15	20
8	3	2	2	1	245	$(\|P\|,\|T\|) = (76,349)$	228.91	16

Table. 7.3. Comparison results for heterogeneous robotic team for the proposed approach

of identical agents. Thus, the current method is evaluated alongside other Discrete Event Systems approaches, suitable for motion planning of homogeneous teams ensuring a global LTL specification. The methods are briefly outlined below. The first two are mathematical programming based on PN approaches, while the third is based on graph-search algorithms.

No. of rob	Algorithm	**(a) Model size**	**(b) Run time** [s]	**(c) Trajectory length**
2	**HLrtPN**	$(\|P\|,\|T\|) = (50,322)$	0.8	33
	WB [35]	$(\|P\|,\|T\|) = (55,288)$	0.86	14
	FB [172]	$(\|P\|,\|T\|) = (13,26), (\|N_B\|,\|T_B\|) = (18,108)$	0.97	14
	TS [50]	$\|N_n\| = 3042$	1.51	14
3	**HLrtPN**	$(\|P\|,\|T\|) = (65,351)$	0.5	28
	WB [35]	$(\|P\|,\|T\|) = (55,288)$	1.1	13
	FB [172]	$(\|P\|,\|T\|) = (13,26), (\|N_B\|,\|T_B\|) = (18,108)$	0.9	13
	TS [50]	$\|N_n\| = 3.9 * 10^3$	1940.33	13
4	**HLrtPN**	$(\|P\|,\|T\|) = (80,380)$	4.5	19
	WB [35]	$(\|P\|,\|T\|) = (55,288)$	0.71	12
	FB [172]	$(\|P\|,\|T\|) = (13,26), (\|N_B\|,\|T_B\|) = (18,108)$	0.76	12
	TS [50]	$\|N_n\| = 5.1 * 10^4$	≈ 3 *days*	–
5	**HLrtPN**	$(\|P\|,\|T\|) = (95,409)$	10.9	17
	WB [35]	$(\|P\|,\|T\|) = (55,288)$	0.74	11
	FB [172]	$(\|P\|,\|T\|) = (13,26), (\|N_B\|,\|T_B\|) = (18,108)$	0.88	11
	TS [50]	$\|N_n\| = 6.6 * 10^5$	–	–
6	**HLrtPN**	$(\|P\|,\|T\|) = (110,438)$	39.5	24
	WB [35]	$(\|P\|,\|T\|) = (55,288)$	0.62	10
	FB [172]	$(\|P\|,\|T\|) = (13,26), (\|N_B\|,\|T_B\|) = (18,108)$	0.88	10
	TS [50]	$\|N_n\| = 8.6 * 10^6$	–	–
7	**HLrtPN**	$(\|P\|,\|T\|) = (125,467)$	133.2	26
	WB [35]	$(\|P\|,\|T\|) = (55,288)$	0.74	9
	FB [172]	$(\|P\|,\|T\|) = (13,26), (\|N_B\|,\|T_B\|) = (18,108)$	1.41	9
	TS [50]	$\|N_n\| = 1.1 * 10^9$	–	–
8	**HLrtPN**	$(\|P\|,\|T\|) = (140,496)$	227.7	16
	WB [35]	$(\|P\|,\|T\|) = (55,288)$	0.17	8
	FB [172]	$(\|P\|,\|T\|) = (13,26), (\|N_B\|,\|T_B\|) = (18,108)$	1.43	8
	TS [50]	$\|N_n\| = 1.4 * 10^{10}$	–	–

Table. 7.4. Comparison results for the homogeneous robotic team between the current, respectively (i), (ii), (iii) methods

Remark 7.3 To maintain the consistency of the comparison procedure, all the mentioned methods, including the current one, are subject to the smallest discrete representation of the

environment w.r.t. the number of partition elements, i.e., one element is associated with a single atomic proposition. Moreover, these methods are integrated into RMTool - MATLAB [119], thus making them accessible for any simulation according to the user's needs.

As previously stated, the team's model size represents one metric taken into consideration for evaluation purposes. Thus, let us express the size of the models for each of the mentioned methods as follows:

- **CPN [35]** - the total number of places and transitions $(|P|,|T|)$ of the defined Composed Petri net model given by the sum of the size of the Petri net model associated with the environment, the size of the Büchi Petri net model associated with the LTL formula, and the number of places for the intermediate layer.
- **FB [172]** - the number of places and transitions of the Petri net model of the environment $(|P|,|T|)$, as well as the size of the Büchi automaton $(|S|,|\rightarrow_B|)$ of the LTL formula (Definition 2.3.2, since both models are examined sequentially.
- **TS [50]** - the total number of nodes in the product automata $|N_n| = |N_{s_r}|^n \times |S|$ considering the size of the transition system for each robot and the size of the Büchi automaton.

Notice that the first two methods have fixed sizes of models regardless of the number of robots in the team versus the last method which is strongly dependent on the size of the team, leading to a state-space explosion that is difficult to maintain for computational operations.

The notation HLrtPN will refer to the proposed method, for an easier visualization in the comparison Table 7.4. Let us introduce the notation $(|P|,|T|)$ to capture the size of the entire model, where $|P|$ and $|T|$ are computed similarly, i.e., $|P| = \sum_{k=1}^{n} |P^{o^k}| + |P^S| + 2$. The result represents the sum of all RobotOPN models for each robot r_k, the size of SpecOPN, and the size of the system net. As defined, the latter representation includes only two places Rb, Ms, and the number of transitions is equal to the number of robots in the team. For this scenario, the size of RobotOPN for r_f is $(15,28)$ (considering one free space place for each floor of the hospital) and $(18,264)$ for the size of SpecOPN due to the automated generation from a Büchi automaton.

Table 7.4 illustrates a comparative study between the current approach and the mentioned relevant Petri net approaches, which embody the defined performance metrics with numerical values. Note that the described methods **(i), (ii), (iii)** do not require multiple simulations for one experiment. Therefore, the **(b) run time** and **(c) trajectory length** are computed only once, without the need to compute an average metric for **(b)** or return the shortest trajectory for **(c)**. The solver used for approaches **(i), (ii), (iii)** is CPLEX [173] for MATLAB.

The **(b) run time** for our proposed work represents the mean time for each experiment, as follows: 1000 simulations for the first three cases (2-5 robots), 250 simulations for 6 robots,

85 simulations for 7 robots and 300 simulations for 8 robots. As observed, the HLrtPN model tends to exhibit steeper increases in running time when more robots are added to the team, compared with **CPN [35], FB [172]**, on account of the number of branches explored by the Renew simulator. The last metric **(c)** portrays the comparison of the trajectory length for the entire robotic team, the smallest value being computed for methods **CPN [35], FB [172]** an account of the optimization problems. In the case of HLrtPN, we expect this metric to decrease for a higher number of simulations for one experiment.

The **TS [50]** model size is computed by a product automata which becomes too large to be computationally tractable for teams of more than 4 robots. Although there are DES methods that are more cost-effective in terms of performances for metrics **(b), (c)** for homogeneous teams, our proposed method is shown to be efficient for heterogeneous teams, due to its flexibility by design, as noted in Table 7.3.

7.3 Experimental validation of the Composed Time Petri net model

Let us end the result chapter by enhancing the flexibility of one of the proposed methods, particularly the *Composed Time Petri net* framework defined in Chapter 4.2. The main idea is to plan high-level trajectories for a robotic system ensuring mission under the MITL formalism and to validate them through experiments. Hence, the theoretical framework considers a team of cobots for an industrial application.

The scenario proposes a solution for a manipulating application dedicated to two cobots, built on the problem tackled in [135] for assembling a cage structure used in constructions. The work is divided into two steps: (a) high-level motion planner based on *Composed Time Petri net* which is tailored accordingly for validating a synchronization mechanism between individual robots' missions; (b) low-level execution planner based on a ROS infrastructure, communicating with the robotic nodes and denoting the pose sequence, allowing the introduction of temporal constraints. This work is one of the few known works in literature to experimentally validate individual MITL missions that are being synchronized in a manipulating application.

Motivation

Focusing on manipulating applications, some works aim to develop collision-free planning algorithms, such as: providing a control law based on a decentralized learning technique [174], proposing a coordination scheme for holonic systems validated by a Colored Petri net model [175], or optimizing the trajectory of the goal pose based on genetic algorithms [176]. A comprehensive workflow for the motion plan is proposed in [177] based on an improved Denavit–Hartenberg method, to prove the correctness of the inverse kinematics, while the angles movements are computed based on a Monte-Carlo method to which a linear trajectory algorithm is added.

These algorithms are essential in the manufacturing industry to ensure a high standard of safety, thus requiring two entities: (i) human knowledge of product development processes, and (ii) the precision and repeatability of operations based on the movement of industrial manipulator robots. To integrate the advantages of both, the robots should be safely operated in the human workspace. Hence, the term *collaborative robots* (known also as *cobots*) is introduced. Several definitions of cobots are provided in the literature, all enhancing the idea of a robot manipulating objects in collaboration with the human operator in a shared workspace [178], e.g., by providing a mixed reality framework facilitating the robot-human interaction [179].

Several studies are concerned with the state-of-the-art of collaborative robotic systems, presenting key challenges such as implementing decision-making methods to provide flexibility and scalability, reducing redundancy, and computing optimal planning strategies, among others [180, 181].

Some examples of applications based on collaborative robots are presented here: in [182] two robots assembled a full-scale vault structure, in [183] an automatic collision-free trajectory was calculated based on a partitioned environment, applying a coverage path planning algorithm to cover the paint on a car, and in [184] two manipulators carry a common payload in an unknown environment, where the roles of leader and helper are dynamically assigned based on robot's performance.

Figure 7.17 presents an overview of the global solution that illustrates the relation between the (a) high-level planner synchronizing the models of the robots based on their MITL missions and (b) low-level planner associated with two robots, as this work provides a real experiment that showcases two robots involved in a manipulating application.

The contributions of the work are reflected in the manipulating application and it includes the following:

- Integration of MITL specifications into real application simulating a manufacturing process between two cobots, based on a tailored model supported by the previously introduced *Composed Time Petri net* framework in [23].
- Proposing a synchronization mechanism between the modified *Composed Time Petri net* representations allocated to each robot, checked in simulations using ROMEO tool [134].
- Implementation of the low-level planner in ROS [185] and validation of the results through real experiments [186].

Problem formulation

Given a robotic manipulator system with $\mathscr{R}$ cobots evolving in a known environment E, the system should automatically build a fixed structure from a set of solid elements based on computed trajectories incorporating constraints in terms of space (regions under set $\mathscr{Y}$, that

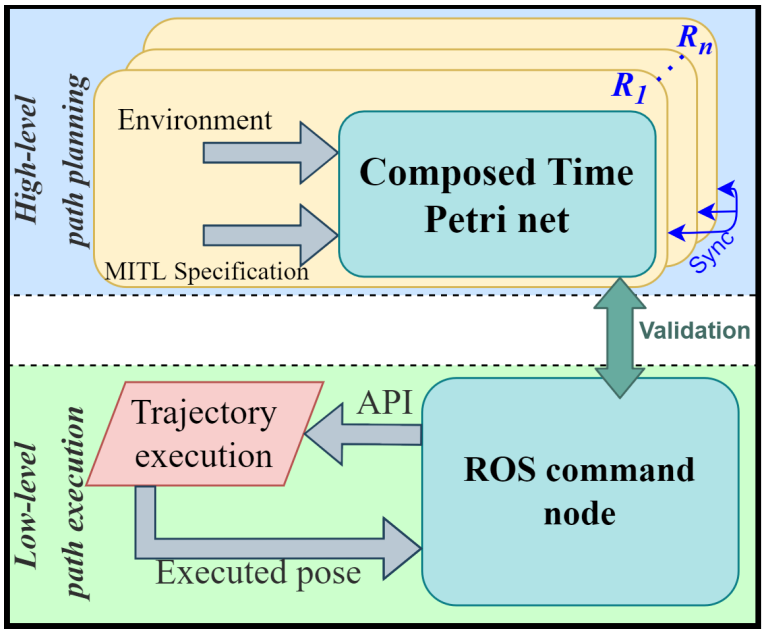


Fig. 7.17. Overview Diagram of the proposed approach

should be reached sequentially and/or synchronously) and time (deadlines for reaching the interest regions). Each cobot receives a set of MITL missions based on a set of actions that are directly linked to the regions of interest. The cobots should cooperate to build the fixed structure, considering the synchronization of actions when needed based on the individual MITL missions. The full motion planning provides preemptive synchronization through the two steps:

- A high-level planning solution is provided by an extension of the introduced *Composed Time Petri net* model (Chapter5), which is built here to provide synchronization between different MITL missions given individually for each robot.
- A low-level robotic framework that ensures the execution of the MITL missions provided by the high-level planning, encapsulating a compensation strategy for the communication delay and an execution time acquisition to validate the mission desired period [186].

The main idea is to use the model *Composed Time Petri net* under the Time Petri net formalism for each cobot that includes the following: one MITL mission φ imposing a set of actions that should be satisfied in given time constraints, and the robotic model encapsulating the capabilities of realizing the actions based on their dependencies with the region of interest $\mathscr{Y}$ that should be reached. The synchronization is achieved formally by coupling the *Composed Time Petri net*s representations for the cobots that need to synchronize, while an experimental plant achieves the validation of the results.

Solution

The solution includes a detailed explanation about both *High-level motion planning* and *Low-level path execution* as observed in Figure 7.17. The examples accompanying the path planning methods proposed for this work concentrate on the scenario that one robot should pick up a tool necessary for the structure that the team of cobots builds.

High-level Motion Planning. The (a) high-level planner from Figure 7.17 is based on the model *Composed Time Petri net*. First, a cell decomposition method is triggered, in order to divide the workspace of the model in a Time Petri net model. Thus, the free space and the regions of interest are captured in the robotic model. Secondly, the given MITL specification is modeled as a Time Petri net mode, as explained in Chapter 4. Since an MITL formula is expressed under a set of atomic propositions that portrays the space requirements of the robot, the coupling of these models is made through a set of places expressing the value of these atomic propositions (*True* or *False*), updated by the robot's movement in the environment. The framework illustrated in the mentioned figure portrays a joint representation of individual Composed Time Petri nets, which are synchronized, through the mechanism defined in Chapter 4.

Chapter 4 details the synchronization mechanism between multiple MITL missions modeled in different *Composed time Petri net* representation, by adding a set of waiting places that are connected to a newly added transition, having its time $[0, 0]$. Since the transition time does not add any new time, but only forces an instant firing when the missions are fulfilled, the expressiveness of the MITL formulae is not altered.

Low-level Path Execution. The (b) low-level planner follows the sequence of points representing the path of the robot ensuring the MITL mission. In order to obtain a planned sequence of points tuned to the time constraints imposed, physical positions in space are user-defined having a time component attached. An MITL mission that consists of a series of these points is subjected to temporal boundaries, resulting in a total period of time at low-level execution.

The execution of the pre-planned sequence of positions consists of a simple and efficient ROS-based architecture that contains individual nodes including API for each robot used.

The ROS nodes coordinate the executions of the individual paths generated by the previously mentioned formalism and delivered through positional commands. The robot's movements and the time of arrival in the right postures are recorded to be analyzed using a similar node. The desired synchronization between the robots is ensured by the position consecutively as a natural consequence of the planning at a similar starting point. The validation of the high-level path planning is quantified by the level of similitude between the time moments presumed in planning and the positional instantaneous moments of arriving in specific postures.

As presented in Figure 7.17, the execution layer contributes as a low-level component created to validate the high-level planning, by allowing a direct correspondence between

the mission and the robot trajectory. The synchronization in the mentioned figure represents a desired moment in time on which both robots reach a desired pose, but it is not a communication signal. Specifically, the *Composed Time Petri net* models ensure through a model-checking method that the MITL missions are satisfied synchronously under the imposed time constraints, maintaining the individual sequencing of the actions, while the low-level planner ensures that the deliberated trajectories follow the ordered actions.

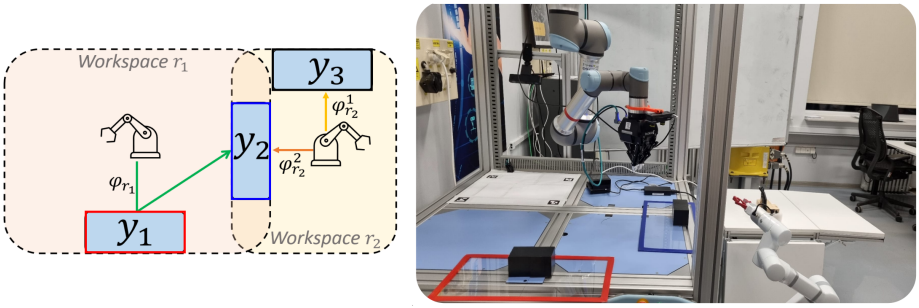
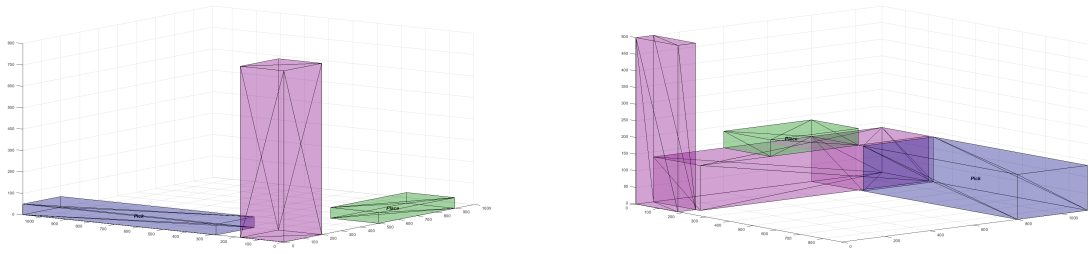


Fig. 7.18. Workspace configuration of the experimental plant: top-view diagram (left side) and side-view real (right side)

Result analysis

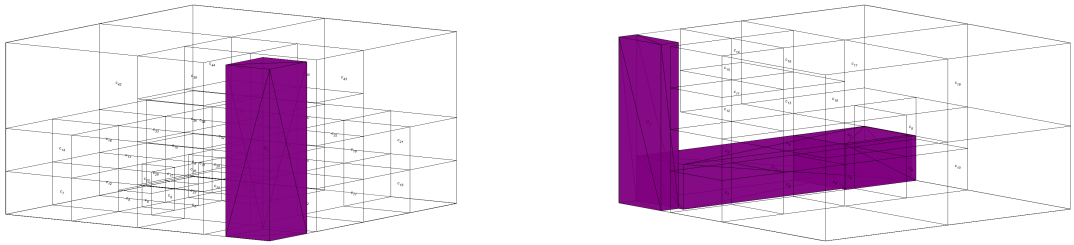
For an easier understanding of the experimental plant, Figure 7.18 illustrates the configuration of the cobots' workspaces and their regions of interest connected with the MITL specifications. Let us consider a cooperative system based on a cobot UR5 from Universal Robots [187] and a cobot LM3 from Lebai Robotics [188], that should build a fixed structure from solid pieces. The cobot UR5 (denoted for r_1 for simplicity) is responsible for picking up the pieces from the dedicated region y_1 (red boundary) and placing them in the construction area y_2 (blue boundary). Cobot LM3 (denoted with r_2) manages the linkage between the solid pieces in the building region y_2, for which the robot should replace its gripper in the region y_3 (black boundary on the left side associated with the white table on the right side) based on the type of linkage procedure that it has to do. The current experiment considers that r_2 should use a welding gun.

Figure 7.19 portrays the workspace modeling in MATLAB of r_1 (UR5) on the left side and r_2 (LM3) on the right side. The 3D polygonal shapes indicate the regions of interest for each cobot, with blue showing the pick regions y_1 (for pieces) of r_1 and y_3 (for welding gun) of r_2, and with green showing the common area y_2 where the pieces are glued together. The purple shapes represent the obstacles in the workspace which are removed from the robot's configuration space as a result of the cell decomposition method (Figure 7.20). For example, r_1 is framed by a fixed structure, visible also in Figure 7.18 on the right side, consisting of a gray column between the cobots. The second cobot has two obstacles, illustrated by the



(a) Representation workspace of cobot r_1 (b) Representation workspace of cobot r_2

Fig. 7.19. Workspaces representations of the cobots, illustrating each relevant region of interest



(a) Partitioned environment of r_1 (b) Partitioned environment of r_2

Fig. 7.20. Cuboid rectangular cell decomposition of the workspaces of each cobot

same gray column, as well as the lower space under the first cobot, since r_2 is smaller and has another space allowed for the movement.

Figure 7.20 illustrates the partitioned environment for each robot considered for this application. the cell decomposition technique is based on the 3D method described by Algorithm 1 in Chapter 2.1, considering the precision $\varepsilon = 4$ (the maximum number of division for each axis). Notice that the robot is absent in the workspace to emphasize the cells resulting from the cell decomposition method, mapping the free space.

The MITL specifications for r_1, respectively r_2 are denoted with φ_{r_1}, respectively $\varphi^1_{r_2}$ and $\varphi^2_{r_2}$, and are given as follows:

$$\varphi_{r_1} = \Diamond_{\tau_1} Idle_{r_1} \wedge (Idle_{r_1} \rightarrow \Diamond_{\tau_2} PickPiece) \wedge (PickPiece \rightarrow \Diamond_{\tau_3} PlacePiece) \tag{7.3}$$

$$\begin{aligned} \varphi^1_{r_2} &= \Diamond_{\tau_1} Idle_{r_2} \wedge (Idle_{r_2} \rightarrow \Diamond_{\tau_2} PickTool) \wedge (PickTool \rightarrow \Diamond_{\tau_3} Idle_{r_2}) \\ \varphi^2_{r_2} &= \Diamond_{\tau_4} UseTool \wedge (UseTool \rightarrow \Diamond_{\tau_3} Idle_{r_2}) \end{aligned} \tag{7.4}$$

The equation 7.3 describes the mission of robot r_1, indicating first reaching with the robot's end effector its intermediary pose $Idle_{r_1}$ represented by fixed coordinates, considering

the time upper bound τ_1. Notice that $Idle$ represents a controlled pose for a cobot ensuring the synchronized start of the system from a known state. Moreover, this acts as an off-duty robot posture before beginning each sequence of their actions. Immediately after the $Idle$ pose is reached, the second action of the robot is to pick up a piece *PickPiece* under τ_2 time units, the task being connected directly with the region y_1. The last requirement of the robot is to place the picked piece ensuring the time τ_3.

On the other hand, robot r_2 receives two missions (7.4): $\varphi_{r_2}^1$ requires to pick up a tool *PickTool* in time τ_2 once it reaches its Idle pose $Idle_{r_2}$ in time τ_1, and afterward returning to its Idle pose again in time τ_3; $\varphi_{r_2}^2$ requires to use the tool $UseTool$ under time $\tau_4 = \tau_1 + \tau_2$. The time requirements are with respect to each other, in the sense that the clock for the time interval $[0, \tau_2]$ starts once the previous action is achieved.

The design decision for assigning two MITL specifications to the robot r_2 includes an easier modularization of the actions to allow reproducibility with respect to the considered application, where the actions of pick-and-place a piece, respectively, using the tool repeat several times in the construction process.

The synchronizations of the robots considering their actions under time constraints are visualized in Figure 7.21. This diagram portrays the sequence of the actions of each robot, and their synchronizations, e.g., for r_1, the *PickPiece* operation is ensured after the robot reaches the $Idle_{r_1}$ pose. In addition, the Idle pose is reached by both robots at the same time, after τ_1 time. Let us consider that the cobots are building a structure from $n \in \mathbf{N}$ pieces.

The high-level trajectories are ensured in two phases. Firstly, the MITL specification φ_{r_1} (green boundary) is coordinated with MITL mission $\varphi_{r_2}^1$ (yellow boundary), in order to pick up both the first piece and the tool required for the construction of the structure. Secondly, while r_1 places the rest of $n-1$ pieces, the robot r_2 synchronizes its movements through MITL mission $\varphi_{r_2}^2$ (orange boundary). After the last pieces it is placed in the construction area y_2, the second robot operates independently of the tool, thus no synchronization is required for this action.

The experiment considered the following values, which will be further analyzed for the verification of the global algorithm: number of pieces $n = 4$ assumed to be identical, $\tau_1 = 4$ seconds, $\tau_2 = 7$ seconds, $\tau_3 = 7$ seconds and $\tau_4 = 11$ seconds.

(a) High-level path planning. One advantage of the ROMEO tool concerning the modeling of a Time Petri net model, which for us is given by the union of two *Composed Time Petri net*, is emphasized through the model-checking properties. Particularly, the tool allows for checking if a desired marking is reached by returning a sequence of timed transitions. Let us consider two global models: (i) r_1 with MITL φ_{r_1} and for r_2 with MITL $\varphi_{r_2}^1$, respectively; (ii) r_1 with MITL φ_{r_1} and for r_2 with MITL $\varphi_{r_2}^2$.

The first global model consists of 87 places and 63 transitions, while the second model consists of 79 and 57 transitions. In both cases, the total number of places for both cobots is given by the sum between the places considering also the places and transitions needed

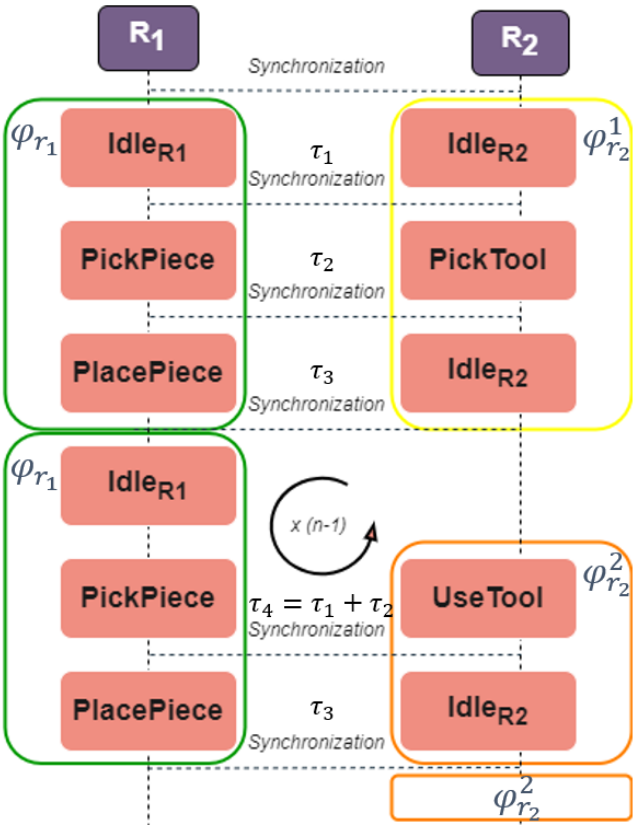
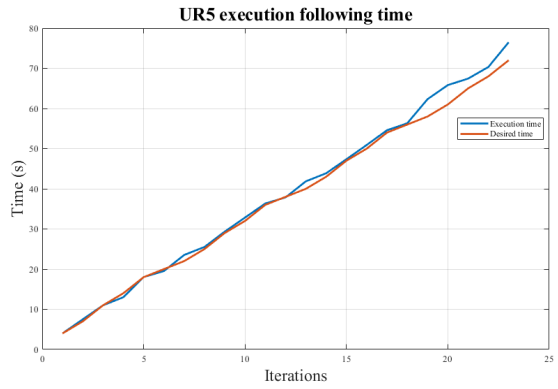


Fig. 7.21. Time Sequence Diagrams for both robots

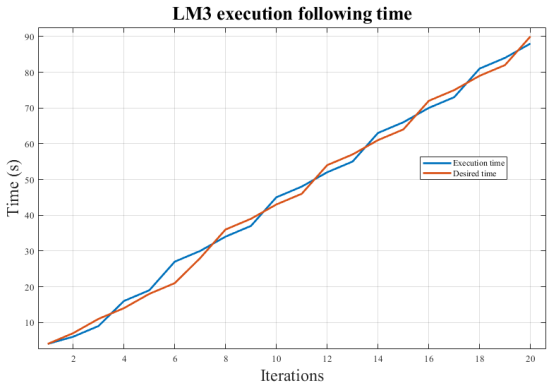
for synchronization. For example, in the first case (i), the places modeling the robots is 17, the places modeling the *True* and *False* value of the atomic propositions is 12, the places modeling the MITL specification is 52, and the waiting places used for synchronization is 6, e.g., 2 places assigned to $Idle_{r_1}$ and $Idle_{r_2}$. The transitions computation is similar, with the run time given by the model-checking properties being 30 seconds in the first case (i), and 38 seconds in the second case (ii). Notice that the tool does not inspect the entire state class graph while searching for a solution, a fact observed in the simulation run time metric.

(b) Low-level path execution. The second phase (low-level execution plan) is conducted on ROS on the Melodic distribution, using specialized APIs for both Universal Robot UR5 and Lebai Robotics LM3 collaborative robotic arms. Each of the system modules has a ROS node, facilitating the communication, interaction, and synchronized execution start.

One major problem of individual time synchronization is network and code execution latency, which is compensated as follows. First, the start time of the execution signal is registered. Secondly, the travel time to the next target pose is altered such that any deviation in the previous movements, referenced to the start time, is subtracted with an error-dependent scaling coefficient from the next one. Although this creates a less accurate time between



(a) UR5 time moments evolution



(b) LM3 time moments evolution

Fig. 7.22. Time-comparison for real-world execution

	UR5			LM3	
	$Idle_{r_1}$	*PickPiece*	*PlacePiece*	$Idle_{r_2}$	*UseTool*
Desired Time (s)	4	7	7	7	11
Execution μ time (s)	4.002	7.1207	7.9993	7.0013	11.0018

Table. 7.5. Desired time per action vs real-world execution time

missions it is a compromise that eliminates the error accumulation in the total execution time. Without this approach, the execution diverges from the desired planning strategy.

Let it be noted that the system is specially designed such the robots have no reciprocal communication and no waiting-for-event procedure exists, leading to individual resulted dynamics that can demonstrate the time-planned synchronization. A principal node is conducted to drive the application start and to record the time moments in which the robots are reaching positions planned.

For the current application, using the high-level formalism and the low-level execution presented, the numerical cumulative evolutions from Figure 7.22 describe the comparison between the planning-desired execution time and time moments at which each robot reaches the desired poses. The iterations that are on the X axis represent the consecutive periods of readings for the time collected from the robots, based on executed operations. For example, if we consider the robot r_1 (UR5), then the iterations are expressed by the time when the Idle pose is reached ($Idle_{r_1}$), or by the recorded operations of opening and closing the gripper, as part of the *PickPiece* action.

As can be observed, the time moments reported by the robot's physical execution of poses (blue) can be observed closely following the time sequence used in the planning procedure (red). Considering the numerical values of the time restrictions $\tau_i, i = \overline{1,4}$ given in

the MITL specifications and the time sequence diagram from 7.21, the total execution time for LM3, respectively UR5 is 88.01, respectively 76.48 seconds compared with the desired total time of 90, respectively 72 seconds. The dissimilarities resulting from latency are well compensated within a small margin of error fulfilling successfully the MITL specifications.

For a numerical representation of the execution time, a comparison between the desired time (defining the MITL temporal constraints), and the executed period of time, can be found in Table 7.5. The execution μ time represents the mean period for all the repetitions of a similar action, e.g., *PickPlace*. As can be observed from the time evolutions in Figure 7.22 and the comparison Table 7.5, for the position sequence generated and experimented, the cumulative time error is infinitesimal based on the presented method.

A complete real-world execution of the application described in the current work can be found in [186], the execution for which the time comparison is presented in Figure 7.22.

Chapter 8

Concluding remarks

This thesis explored several challenges and solutions regarding the planning strategies for multi-robot systems ensuring a mission, expressed through the temporal logic formalism. Specifically, these planning methods aim to use the advantages of Discrete Event System (DES) representations including an easier handling of the robotic model and facilitating the integration with the model of a temporal logic specification. Starting from a collision-free trajectory for a single robot system, the planning methods evolved towards multi-robotic systems ensuring rich and complex missions. Thus, spatial and temporal constraints towards a set of regions of interest from the workspace, that should be reached and/or avoided by the robotic team, are encoded in this thesis through the temporal logic specification such as Linear Temporal Logic (LTL), and Metric Interval Temporal Logic (MITL).

The planning solutions addressed to the robotic field aim to emphasize the coordination of the robots when task allocation approaches are tackled, scalability concerning the number of the robots in the team, and the versatility of the proposed planning methods towards identical, respectively heterogeneous robotic teams, also considering the applicability of the solutions in futuristic health care domain, an industrial application using cobots and in the agriculture field using UAV team-based.

Chapter 2 introduces the fundamental notions upon which the robotic planning strategies are built. Among the Discrete Event System representations used in the proposed methods, let us recall the Transition Systems, Petri net models, and Time Petri net models. The representations are associated with the robotic team, based on a partitioning technique (2D and 3D cell decomposition) of the workspace leasing to easier handling. On the same note, three types of missions were defined, starting from the Boolean specification expressing the regions of interest that should be visited and/or bypassed throughout the trajectories, towards encoding sequencing and synchronization for the spatial restrictions in the Linear Temporal Logic specification and closing with the specification under Metric Interval Temporal Logic that includes temporal constraints. These formalisms are associated with

an automata model, for which model-checking approaches can be applied to verify if the mission is satisfied by the motion of the robots.

The next chapters of this thesis provide detailed explanations of the proposed frameworks under Discrete Event Systems formalism guaranteeing high-level objectives for robotic teams. Thus, the concluding remarks comprise a set of concise contributions to the field of robotics, upon which future research directions are envisioned.

8.1 Contributions

Two task allocation strategies are presented in **Chapter 3**, particularly based on a task decomposition technique for a given global LTL mission resulting in a set of smaller and independent tasks that leads to a complexity reduction of the solution space, and based on a reallocation task technique starting from a set of trajectories already computed and enforcing the robots for a parallel movement. The latter method is based on the Banker's algorithm suitable for resource allocation problems. Both methods rely on the Petri net model associated with the motion of the robots, a model that portrays a fixed topology when the number of identical robots evolving in the same workspace is increasing or decreasing.

The aim of **Chapter 4** is to propose a novel framework by joining the advantages of the Petri net model, particularly a reduced model known under the term quotient, assigned for the movement of the robotic team and a Büchi automata assigned to the global LTL mission. For this, an intermediate layer of places is added, where each place is associated with an atomic proposition assigned to a region of interest, a set upon which the mission is built. The planning strategy is computed by two Mixed Integer Linear Programming (MILP) problems, one considering the newly defined *Composed Petri net* model, respectively projecting the solution based on the reduced model into the full Petri net model of the workspace. This framework enhances a direct collision-free planning approach compared to other methods from literature based on iterative approaches, through the use of global information on the state of the robotic team indicated by the intermediate layer of places. Moreover, this work has also been extended towards the MITL missions, considering a newly defined model denoted *Composed Time Petri net* in **Chapter 5**. Thus, time constraints are encapsulated in this representation and validated in both simulations and experiment results, considering an industrial application for a team of cobots ensuring individual MITL missions.

Another novel framework is introduced in **Chapter 6**, denoted *High-Level robotic team Petri net* model under the Nets-within-Nets (NwN) paradigm. Based on the state-of-the-art, this approach could be considered a breakthrough, since the NwN paradigm has not been tackled previously to plan trajectories for a heterogeneous robotic team ensuring a global mission, specifically using an LTL formula. The benefits of the NwN formalism are usually highlighted in industrial applications since this formalism incorporates a hierarchical structure of the Petri nets: object nets associated with the local information of a robot state

and a system net associated with the global information about the entire robotic system. Notable, the tokens in the system net are encoded as object nets.

The contribution of this work is to design an object net for each type of robot since the robotic team is heterogeneous, defined *RobotOPN*, respectively designing an object net for the global specification defined *SpecOPN*. The coordination between the object nets is realized through a synchronization function that ensures that the movement of the robots fulfills the mission. The proposed framework has been analyzed through simulations for a team of up to 10 heterogeneous robots and the results were compared with other Discrete Event System methods from literature, concluding in a smaller run time for a large number of simulations (up to 1000 simulations per one experiment).

The contributions that have been disseminated during the PhD research are enumerated as follows:

1. **Four journal papers:**

   (a) **Sofia Hustiu**, Cristian Mahulea, Marius Kloetzer, and Jean-Jacques Lesage. On multi-robot path planning based on Petri net models and LTL specifications. In *IEEE Transactions on Automatic Control, vol. 69, no. 9, pp. 6373-6380,* 2024.

   (b) **Sofia Hustiu**, Ioana Hustiu, Marius Kloetzer, and Cristian Mahulea. LTL task decomposition for 3D high-level path planning. In *Journal of Control Engineering and Applied Informatics, 23(3), pp.76-87*, 2021.

   (c) **Sofia Hustiu**, Eva Robillard, Joaquín Ezpeleta, Cristian Mahulea, and Marius Kloetzer. Multi-robot Motion Planning based on Nets-within-Nets Modeling and Simulation. *Under review*. Available in [Online]: https://arxiv.org/abs/2304.08772, 2023.

   (d) **Sofia Hustiu**. Prerequisites to Design a Collision Free Trajectory in a 3D Dynamic Environment for an UAV. In *Bulletin of the Polytechnic Institute of Iași. Electrical Engineering, Power Engineering, Electronics Section*, 67(2), pp.65-78, 2021.

2. **Six conference proceedings:**

   (a) **Sofia Hustiu**, Alexandru-Florian Brasoveanu, and Andrei-Iulian Iancu. Integration of MITL for Cobots Workflow in a Manipulating Application. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8, 2024.

   (b) **Sofia Hustiu**, Dimos V. Dimarogonas, Cristian Mahulea, and Marius Kloetzer. Multi-robot Motion Planning under MITL Specifications based on Time Petri Nets. In *2023 European Control Conference (ECC) (pp. 1-8). IEEE*, 2023.

(c) **Sofia Hustiu**, Cristian Mahulea, and Marius Kloetzer. Parallel motion execution and path rerouting for a team of mobile robots. In IFAC-PapersOnLine, 55(28), 73–78. In *16th IFAC Workshop on Discrete Event Systems WODES,* 2022.

(d) **Sofia Hustiu**, Marius Kloetzer, Eva Robillard, Alejandro López-Martínez, and Cristian Mahulea. Whitening of greenhouse's roof using drones and Petri net models. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, 2022.

(e) **Sofia Hustiu**, Marius Kloetzer, and Cristian Mahulea. Mission assignment and 3D path planning for a team of UAVs. In 2021 *25th International Conference on System Theory, Control and Computing (ICSTCC)* (pp. 401-406). IEEE, 2021.

(f) **Sofia Hustiu**, Marius Kloetzer, and Adrian Burlacu. Collision Free Path Planning for Unmanned Aerial Vehicles in Environments with Dynamic Obstacles. In 2020 *24th International Conference on System Theory, Control and Computing (ICSTCC)* (pp. 520-525). IEEE, 2020.

8.2 Future research directions

The limitations of the proposed frameworks can be encapsulated in the following challenges: (i) enabling open-source implementation for all the proposed frameworks, (ii) reducing the computational complexity when the number of robots increases, (iii) strengthening the Nets-within-Nets planning approach by adding characteristics modeling the dynamic of the robots and aiming to fulfill the mission under time constraints, and (iv) motion plan collision-free trajectories in unknown environments. Potential future research directions addressing these challenges could be built upon the defined Discrete Event Systems throughout the thesis.

The first two points (i) and (ii) are related to one of the contributions of this thesis, mainly the implementation in the RMTool, as mentioned previously. Firstly, since the *Composed Petri net* model is deployed and available in the RMTool MATLAB, this representation could be extended toward partially unknown environments. The RMTool includes a feature of defining partially unknown environments for individual robots and planning trajectories for Boolean specifications. Thus, the RMTool facilitates a simulation workspace for implementing the idea of modeling uncertain environments under Petri net formalism, combined with an intermediate layer of places providing a probabilistic view of the global state of the robotic team, together with the Büchi automata for the LTL mission.

Similarly, the RMTool could be extended towards MITL specifications, since is currently providing only Boolean and LTL missions to be ensured by the robotic team. Therefore, the representation noted *Composed Time Petri net* could be automatically built for each robot and a planning algorithm could be developed for a more controlled solution search in contrast with the model-checking approach that has been used so far for this work.

The third challenge (iii), concerning the *High-Level robotic team Petri net* model under the Nets-within-Nets paradigm is also accessible to researchers, enhancing its potential of being further tailored towards planning under time constraints. An interesting approach would be to combine the high-level benefits of this framework with the low-level path execution, through the addition of control algorithms. One idea is to encode a control method when a transition is fired and executed by a robot, specifying the motion action considering the robot's dynamics of moving from the free space towards a region of interest.

Lastly, planning a robotic team in unknown environments is crucial (iv), especially in search-and-rescue scenarios, where the robots should explore the environment in a coordinated manner without colliding with obstacles. Here, several directions could be addressed. By starting modeling the robotic system in a probabilistic mode, the robots could be allowed to provide information about the environment based on their distance towards the regions of interest. Another idea might investigate the modeling of uncertainty through the mission that should be ensured, such as Probabilistic Temporal Logic formalism.

For all the proposed models, the adaptability and robustness of the frameworks could also be improved by expanding them towards distributed systems and applying mathematical programming algorithms to efficiently reduce computational solving time.

Chapter 9

Observații finale

Această teză a explorat mai multe provocări și soluții privind strategiile de planificare pentru sisteme multi-robot destinate îndeplinirii unei misiuni, exprimate prin formalismul logicii temporale. În mod specific, aceste metode de planificare urmăresc să utilizeze avantajele reprezentărilor Sistemelor de Evenimente Discrete (SED), inclusiv o gestionare mai simplă a modelului robotic și facilitarea integrării cu modelul unei specificații de logică temporală. Pornind de la o traiectorie fără coliziuni pentru un sistem robotic individual, metodele de planificare au evoluat către sisteme multi-robot capabile să asigure misiuni complexe și bogate. Astfel, constrângerile spațiale și temporale asociate unui set de regiuni de interes din spațiul de lucru, care trebuie atinse și/sau evitate de echipa robotică, sunt codificate în această teză prin specificațiile logicii temporale precum Logica Temporală Liniară (LTL) și Logica Temporală Metrică pe Interval (MITL).

Soluțiile de planificare adresate domeniului roboticii își propun să evidențieze coordonarea roboților în abordările de alocare a sarcinilor, scalabilitatea în raport cu numărul de roboți din echipă și versatilitatea metodelor propuse de planificare pentru echipe robotice identice sau eterogene, luând în considerare, de asemenea, aplicabilitatea soluțiilor în domenii futuriste precum asistența medicală, o aplicație industrială utilizând cobots (roboți colaborativi) și în agricultură folosind echipe de UAV-uri.

Capitolul 2 introduce noțiunile fundamentale pe care sunt construite strategiile de planificare robotică. Printre reprezentările Sistemelor de Evenimente Discrete utilizate în metodele propuse, amintim Sistemele de Tranziție, modelele de Rețea Petri și modelele de Rețea Petri Temporală. Reprezentările sunt asociate echipei robotice, bazându-se pe o tehnică de partiționare (decompozitie celulară 2D și 3D) a spațiului de lucru, facilitând astfel gestionarea. În același sens, au fost definite trei tipuri de misiuni, pornind de la specificația Booleană care exprimă regiunile de interes ce trebuie vizitate și/sau evitate de-a lungul traiectoriilor, continuând prin a codifica secvențierea și sincronizarea restricțiilor spațiale în specificațiile LTL și finalizând cu specificațiile MITL care includ constrângeri temporale. Aceste formalizări sunt asociate unui model de automat pentru care pot fi aplicate metode de

verificare a modelului (model-checking) pentru a verifica dacă misiunea este îndeplinită de mișcarea roboților.

Capitolele următoare ale tezei oferă explicații detaliate ale cadrelor propuse sub formalismul Sistemelor de Evenimente Discrete, garantând obiective de nivel înalt pentru echipele robotice. Astfel, concluziile includ un set de contribuții concise pentru domeniul roboticii, pe baza cărora se prevăd direcții viitoare de cercetare.

9.1 Contribuții

Două strategii de alocare a sarcinilor sunt prezentate în **Capitolul 3**, în mod particular bazate pe o tehnică de decompozitie a unei misiuni globale LTL într-un set de sarcini mai mici și independente, ceea ce duce la reducerea complexității spațiului soluțiilor, și pe o tehnică de realocare a sarcinilor, pornind de la un set de traiectorii deja calculate și forțând roboții să se miște în paralel. Aceasta din urmă se bazează pe algoritmul Bancherului, potrivit pentru probleme de alocare a resurselor (reprezentate de spațiul liber de evoluție al roboților). Ambele metode se bazează pe modelul Rețelei Petri asociat mișcării roboților, un model care menține o topologie fixă atunci când numărul de roboți identici care evoluează în același spațiu de lucru crește sau scade.

Scopul **Capitolului 4** este de a propune un cadru nou prin combinarea avantajelor modelului Rețelei Petri, în mod special un model redus cunoscut sub denumirea de quotient, asociat mișcării echipei robotice, și a unui automat Büchi asociat misiunii globale LTL. Pentru aceasta, se adaugă un strat intermediar de locuri, unde fiecare loc este asociat cu o propoziție atomică atribuită unei regiuni de interes, un set pe baza căruia este construită misiunea. Strategia de planificare este calculată prin două probleme de Programare Liniară cu Numere Întregi Mixte (MILP), una considerând modelul nou definit de *Rețea Petri Compusă* și proiectând soluția bazată pe modelul redus în modelul complet al Rețelei Petri al spațiului de lucru (**Capitolul 5**).

Un alt cadru inovator este introdus în **Capitolul 6**, denumit *modelul de Rețea Petri pentru o echipă robotică de nivel înalt* în cadrul paradigmei *Nets-within-Nets* (NwN). Pe baza stadiului actual al cercetărilor, această abordare ar putea fi considerată un progres, deoarece paradigma NwN nu a fost explorată anterior pentru planificarea traiectoriilor unei echipe robotice eterogene care să asigure o misiune globală, utilizând în mod specific o formulă LTL.

În cadrul acestei contribuții, se dorește proiectarea unei rețele de obiecte pentru fiecare tip de robot, având în vedere că echipa de roboți este eterogenă, definită ca *RobotOPN*, respectiv proiectarea unei rețele de obiecte pentru specificația globală, definită ca *SpecOPN*. Coordonarea între rețelele de obiecte este realizată printr-o funcție de sincronizare care asigură că mișcarea roboților îndeplinește misiunea. Cadrul propus a fost analizat prin simulări pentru o echipă formată din până la 10 roboți eterogeni, iar rezultatele au fost

comparate cu alte metode din literatura de specialitate bazate pe Sisteme de Evenimente Discrete, concluzionându-se un timp de rulare mai mic pentru un număr mare de simulări (până la 1000 de simulări per experiment).

Contribuțiile acestei teze au fost diseminate astfel:

1. **Patru articole în reviste:**

   (a) **Sofia Hustiu**, Cristian Mahulea, Marius Kloetzer, and Jean-Jacques Lesage. On multi-robot path planning based on Petri net models and LTL specifications. In *IEEE Transactions on Automatic Control, vol. 69, no. 9, pp. 6373-6380,* 2024.

   (b) **Sofia Hustiu**, Ioana Hustiu, Marius Kloetzer, and Cristian Mahulea. LTL task decomposition for 3D high-level path planning. In *Journal of Control Engineering and Applied Informatics, 23(3), pp.76-87*, 2021.

   (c) **Sofia Hustiu**, Eva Robillard, Joaquín Ezpeleta, Cristian Mahulea, and Marius Kloetzer. Multi-robot Motion Planning based on Nets-within-Nets Modeling and Simulation. *Under review*. Available in [Online]: https://arxiv.org/abs/2304.08772, 2023.

   (d) **Sofia Hustiu**. Prerequisites to Design a Collision Free Trajectory in a 3D Dynamic Environment for an UAV. In *Bulletin of the Polytechnic Institute of Iași. Electrical Engineering, Power Engineering, Electronics Section*, 67(2), pp.65-78, 2021.

2. **Șase lucrări de conferință:**

   (a) **Sofia Hustiu**, Alexandru-Florian Brasoveanu, and Andrei-Iulian Iancu. Integration of MITL for Cobots Workflow in a Manipulating Application. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8, 2024.

   (b) **Sofia Hustiu**, Dimos V. Dimarogonas, Cristian Mahulea, and Marius Kloetzer. Multi-robot Motion Planning under MITL Specifications based on Time Petri Nets. In *2023 European Control Conference (ECC) (pp. 1-8). IEEE*, 2023.

   (c) **Sofia Hustiu**, Cristian Mahulea, and Marius Kloetzer. Parallel motion execution and path rerouting for a team of mobile robots. In IFAC-PapersOnLine, 55(28), 73–78. In *16th IFAC Workshop on Discrete Event Systems WODES,* 2022.

   (d) **Sofia Hustiu**, Marius Kloetzer, Eva Robillard, Alejandro López-Martínez, and Cristian Mahulea. Whitening of greenhouse's roof using drones and Petri net models. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, 2022.

(e) **Sofia Hustiu**, Marius Kloetzer, and Cristian Mahulea. Mission assignment and 3D path planning for a team of UAVs. In 2021 *25th International Conference on System Theory, Control and Computing (ICSTCC)* (pp. 401-406). IEEE, 2021.

(f) **Sofia Hustiu**, Marius Kloetzer, and Adrian Burlacu. Collision Free Path Planning for Unmanned Aerial Vehicles in Environments with Dynamic Obstacles. In 2020 *24th International Conference on System Theory, Control and Computing (ICSTCC)* (pp. 520-525). IEEE, 2020.

9.2 Direcții viitoare de cercetare

Limitările cadrelor propuse pot fi sintetizate în următoarele provocări: (i) implementarea open-source pentru toate cadrele propuse, (ii) reducerea complexității computaționale pe măsură ce numărul roboților crește, (iii) consolidarea abordării de planificare Nets-within-Nets prin adăugarea caracteristicilor care modelează dinamica roboților, având ca scop îndeplinirea misiunii sub constrângeri de timp, și (iv) planificarea de traiectorii fără coliziuni în medii necunoscute. Direcțiile potențiale de cercetare viitoare care să abordeze aceste provocări s-ar putea baza pe Sistemele de Evenimente Discrete definite pe parcursul acestei teze.

Primele două puncte (i) și (ii) sunt legate de una dintre contribuțiile acestei teze, în principal implementarea în RMTool, așa cum s-a menționat anterior. În primul rând, având în vedere că modelul de *Rețea Petri Compusă* este implementat și disponibil în RMTool MATLAB, această reprezentare ar putea fi extinsă către medii parțial necunoscute. RMTool include o funcționalitate pentru definirea mediilor parțial necunoscute pentru roboți individuali și planificarea traiectoriilor pentru specificații Booleene. Astfel, RMTool facilitează un spațiu de simulare pentru implementarea ideii de modelare a mediilor incerte sub formalismul Rețelelor Petri, combinat cu un strat intermediar de locuri care oferă o viziune probabilistică asupra stării globale a echipei robotice, împreună cu automatul Büchi pentru misiunea LTL.

În mod similar, RMTool ar putea fi extins pentru specificații MITL, deoarece în prezent oferă doar misiuni Booleene și LTL pentru a fi îndeplinite de echipa robotică. Prin urmare, reprezentarea denumită *Rețea Petri Temporală Compusă* ar putea fi construită automat pentru fiecare robot, iar un algoritm de planificare ar putea fi dezvoltat pentru o căutare a soluțiilor mai controlată, în contrast cu abordarea de verificare a modelului utilizată până acum în acest context.

A treia provocare (iii), referitoare la *modelul de Rețea Petri pentru o echipă robotică de nivel înalt* în cadrul paradigmei Nets-within-Nets, este, de asemenea, accesibilă cercetătorilor, ceea ce sporește potențialul de a fi adaptată pentru planificarea sub constrângeri de timp. O abordare interesantă ar fi combinarea beneficiilor de nivel înalt ale acestui cadru cu execuția traiectoriilor de nivel jos, prin adăugarea unor algoritmi de control. O idee ar fi codificarea unei metode de control atunci când o tranziție este activată și executată de un

robot, specificând acțiunea de mișcare luând în considerare dinamica robotului în mișcarea din spațiul liber către o regiune de interes.

În cele din urmă, planificarea unei echipe de roboți în medii necunoscute este crucială (iv), în special în scenarii de căutare și salvare, unde roboții ar trebui să exploreze mediul în mod coordonat fără să se ciocnească de obstacole. Aici pot fi abordate mai multe direcții. Pornind de la modelarea sistemului robotic într-un mod probabilistic, roboții ar putea oferi informații despre mediu bazate pe distanța lor față de regiunile de interes. O altă idee ar putea investiga modelarea incertitudinii prin misiunea care trebuie îndeplinită, cum ar fi formalismul Logicii Temporale Probabilistice.

Pentru toate modelele propuse, adaptabilitatea și robustețea cadrelor ar putea fi, de asemenea, îmbunătățite prin extinderea acestora către sisteme distribuite și aplicarea algoritmilor de programare matematică pentru a reduce eficient timpul de rezolvare computațională.

Chapter 10

Resumen y conclusiónes

Esta tesis exploró varios desafíos y soluciones relacionados con las estrategias de planificación para sistemas multi-robot que garantizan una misión, expresada mediante el formalismo de lógica temporal. Específicamente, estos métodos de planificación tienen como objetivo aprovechar las ventajas de las representaciones del Sistema de Eventos Discretos (DES), incluyendo un manejo más sencillo del modelo robótico y la facilitación de la integración con el modelo de una especificación de lógica temporal. Partiendo de una trayectoria libre de colisiones para un sistema de un solo robot, los métodos de planificación evolucionaron hacia sistemas multi-robot que aseguran misiones ricas y complejas. Así, las restricciones espaciales y temporales hacia un conjunto de regiones de interés del espacio de trabajo, que deben ser alcanzadas y/o evitadas por el equipo robótico, se codifican en esta tesis a través de especificaciones de lógica temporal como la Lógica Temporal Lineal (LTL) y la Lógica Temporal Métrica de Intervalos (MITL).

Las soluciones de planificación dirigidas al campo robótico tienen como objetivo enfatizar la coordinación de los robots cuando se abordan enfoques de asignación de tareas, la escalabilidad en relación con el número de robots en el equipo y la versatilidad de los métodos de planificación propuestos para equipos robóticos idénticos y heterogéneos. Además, se considera la aplicabilidad de las soluciones en dominios futuristas como el cuidado de la salud, aplicaciones industriales utilizando cobots y en el campo de la agricultura utilizando equipos de UAV.

El **Capítulo 2** introduce las nociones fundamentales sobre las cuales se construyen las estrategias de planificación robótica. Entre las representaciones del Sistema de Eventos Discretos utilizadas en los métodos propuestos, se incluyen los Sistemas de Transición, los modelos de redes de Petri y los modelos de redes de Petri Temporizadas. Estas representaciones están asociadas al equipo robótico, basándose en una técnica de particionamiento (descomposición celular 2D y 3D) del espacio de trabajo, facilitando su manejo. En la misma línea, se definieron tres tipos de misiones, comenzando con la especificación booleana que expresa las regiones de interés que deben ser visitadas y/o evitadas a lo largo de las

trayectorias, pasando por la codificación de secuencias y sincronización para las restricciones espaciales en la especificación de Lógica Temporal Lineal, y concluyendo con la especificación bajo Lógica Temporal Métrica de Intervalos, que incluye restricciones temporales. Estos formalismos están asociados con un modelo de autómatas, para el cual se pueden aplicar enfoques de verificación formal (model-checking) para verificar si la misión es satisfecha por el movimiento de los robots.

Los capítulos siguientes de esta tesis proporcionan explicaciones detalladas de los marcos propuestos bajo el formalismo de los Sistemas de Eventos Discretos, garantizando objetivos de alto nivel para equipos robóticos. Por lo tanto, las conclusiones comprenden un conjunto de contribuciones concisas al campo de la robótica, sobre las cuales se vislumbran futuras líneas de investigación.

10.1 Contribuciones

Dos estrategias de asignación de tareas se presentan en **el Capítulo 3**, basadas particularmente en una técnica de descomposición de tareas para una misión global LTL dada, resultando en un conjunto de tareas más pequeñas e independientes que llevan a una reducción de la complejidad del espacio de soluciones, y en una técnica de reasignación de tareas a partir de un conjunto de trayectorias ya calculadas, forzando a los robots a un movimiento paralelo. Este último método se basa en el algoritmo del Banquero, adecuado para problemas de asignación de recursos. Ambos métodos dependen del modelo de Redes de Petri asociado al movimiento de los robots, un modelo que refleja una topología fija cuando el número de robots idénticos que evolucionan en el mismo espacio de trabajo aumenta o disminuye.

El objetivo de **el Capítulo 4** es proponer un nuevo marco que combina las ventajas del modelo de Redes de Petri, particularmente un modelo reducido conocido como "cociente", asignado para el movimiento del equipo robótico, y un autómata de Büchi asignado a la misión global LTL. Para esto, se agrega una capa intermedia de lugares, donde cada lugar está asociado a una proposición atómica asignada a una región de interés, un conjunto sobre el cual se construye la misión. La estrategia de planificación se calcula mediante dos problemas de Programación Lineal Entera Mixta (MILP), uno considerando el modelo recién definido de *Red de Petri Compuesta*, y proyectando la solución basada en el modelo reducido hacia el modelo completo de Red de Petri del espacio de trabajo. Este marco mejora un enfoque de planificación directa libre de colisiones en comparación con otros métodos de la literatura basados en enfoques iterativos, utilizando la información global sobre el estado del equipo robótico indicada por la capa intermedia de lugares. Además, este trabajo se ha extendido a misiones MITL, considerando un modelo recién definido denominado *Red de Petri Temporal Compuesta* (**el Capítulo 5**). Así, las restricciones temporales se encapsulan en esta representación y se validan tanto en simulaciones como en resultados experimentales,

considerando una aplicación industrial para un equipo de cobots que garantizan misiones MITL individuales.

Otro marco novedoso es introducido en **el Capítulo 6**, denominado *Modelo de Red de Petri de Equipo Robótico de Alto Nivel* bajo el paradigma *Nets-within-Nets* (NwN). Según el estado del arte, este enfoque podría considerarse un avance, ya que el paradigma NwN no ha sido previamente abordado para planificar trayectorias de un equipo robótico heterogéneo garantizando una misión global, específicamente utilizando una fórmula LTL. Los beneficios del formalismo NwN suelen destacarse en aplicaciones industriales, ya que este formalismo incorpora una estructura jerárquica de Redes de Petri: redes objeto asociadas con la información local del estado de un robot y una red del sistema asociada con la información global sobre todo el sistema robótico. Es notable que los tokens en la red del sistema se codifican como redes objeto.

La contribución de este trabajo es diseñar una red objeto para cada tipo de robot, definida como *RobotOPN*, y diseñar una red objeto para la especificación global, definida como *SpecOPN*. La coordinación entre las redes objeto se realiza mediante una función de sincronización que garantiza que el movimiento de los robots cumpla la misión. El marco propuesto ha sido analizado a través de simulaciones para un equipo de hasta 10 robots heterogéneos y los resultados se compararon con otros métodos de Sistemas de Eventos Discretos de la literatura, concluyendo con un menor tiempo de ejecución en un gran número de simulaciones (hasta 1000 simulaciones por experimento).

Las contribuciones que se han divulgado durante la investigación del doctorado se enumeran a continuación:

1. **Cuatro artículos de revista:**

   (a) **Sofia Hustiu**, Cristian Mahulea, Marius Kloetzer, and Jean-Jacques Lesage. On multi-robot path planning based on Petri net models and LTL specifications. In *IEEE Transactions on Automatic Control, vol. 69, no. 9, pp. 6373-6380,* 2024.

   (b) **Sofia Hustiu**, Ioana Hustiu, Marius Kloetzer, and Cristian Mahulea. LTL task decomposition for 3D high-level path planning. In *Journal of Control Engineering and Applied Informatics, 23(3), pp.76-87*, 2021.

   (c) **Sofia Hustiu**, Eva Robillard, Joaquín Ezpeleta, Cristian Mahulea, and Marius Kloetzer. Multi-robot Motion Planning based on Nets-within-Nets Modeling and Simulation. *Under review*. Available in [Online]: https://arxiv.org/abs/2304.08772, 2023.

   (d) **Sofia Hustiu**. Prerequisites to Design a Collision Free Trajectory in a 3D Dynamic Environment for an UAV. In *Bulletin of the Polytechnic Institute of Iași. Electrical Engineering, Power Engineering, Electronics Section*, 67(2), pp.65-78, 2021.

2. **Seis actas de conferencias:**

   (a) **Sofia Hustiu**, Alexandru-Florian Brasoveanu, and Andrei-Iulian Iancu. Integration of MITL for Cobots Workflow in a Manipulating Application. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8, 2024.

   (b) **Sofia Hustiu**, Dimos V. Dimarogonas, Cristian Mahulea, and Marius Kloetzer. Multi-robot Motion Planning under MITL Specifications based on Time Petri Nets. In *2023 European Control Conference (ECC) (pp. 1-8). IEEE*, 2023.

   (c) **Sofia Hustiu**, Cristian Mahulea, and Marius Kloetzer. Parallel motion execution and path rerouting for a team of mobile robots. In IFAC-PapersOnLine, 55(28), 73–78. In *16th IFAC Workshop on Discrete Event Systems WODES,* 2022.

   (d) **Sofia Hustiu**, Marius Kloetzer, Eva Robillard, Alejandro López-Martínez, and Cristian Mahulea. Whitening of greenhouse's roof using drones and Petri net models. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, 2022.

   (e) **Sofia Hustiu**, Marius Kloetzer, and Cristian Mahulea. Mission assignment and 3D path planning for a team of UAVs. In 2021 *25th International Conference on System Theory, Control and Computing (ICSTCC)* (pp. 401-406). IEEE, 2021.

   (f) **Sofia Hustiu**, Marius Kloetzer, and Adrian Burlacu. Collision Free Path Planning for Unmanned Aerial Vehicles in Environments with Dynamic Obstacles. In 2020 *24th International Conference on System Theory, Control and Computing (ICSTCC)* (pp. 520-525). IEEE, 2020.

10.2 Direcciones de investigación futura

Las limitaciones de los marcos propuestos pueden encapsularse en los siguientes desafíos: (i) habilitar la implementación de código abierto para todos los marcos propuestos, (ii) reducir la complejidad computacional cuando aumenta el número de robots, (iii) fortalecer el enfoque de planificación de Redes dentro de Redes (Nets-within-Nets) añadiendo características que modelen la dinámica de los robots y busquen cumplir la misión bajo restricciones de tiempo, y (iv) planificar trayectorias libres de colisiones en entornos desconocidos. Las posibles direcciones de investigación futura para abordar estos desafíos podrían basarse en los Sistemas de Eventos Discretos definidos a lo largo de esta tesis.

Los dos primeros puntos (i) y (ii) están relacionados con una de las contribuciones de esta tesis, principalmente la implementación en RMTool, como se mencionó previamente. En primer lugar, dado que el modelo de *Red de Petri Compuesta* está implementado y disponible en RMTool MATLAB, esta representación podría extenderse hacia entornos parcialmente

desconocidos. RMTool incluye una función para definir entornos parcialmente desconocidos para robots individuales y planificar trayectorias para especificaciones Booleanas. Por lo tanto, RMTool facilita un espacio de simulación para implementar la idea de modelar entornos inciertos bajo el formalismo de Redes de Petri, combinado con una capa intermedia de lugares que proporciona una vista probabilística del estado global del equipo robótico, junto con el autómata de Büchi para la misión LTL.

De manera similar, RMTool podría extenderse hacia especificaciones MITL, ya que actualmente solo proporciona misiones Booleanas y LTL para ser cumplidas por el equipo robótico. Por lo tanto, la representación denominada *Red de Petri Temporal Compuesta* podría construirse automáticamente para cada robot y desarrollarse un algoritmo de planificación para una búsqueda de soluciones más controlada en contraste con el enfoque de verificación de modelos que se ha utilizado hasta ahora en este trabajo.

El tercer desafío (iii), relacionado con el *modelo de Red de Petri de equipo robótico de alto nivel* bajo el paradigma de Redes dentro de Redes, también está disponible para los investigadores, lo que aumenta su potencial para ser adaptado a la planificación bajo restricciones de tiempo. Un enfoque interesante sería combinar los beneficios de alto nivel de este marco con la ejecución de trayectorias de bajo nivel, mediante la adición de algoritmos de control. Una idea es codificar un método de control cuando una transición se activa y es ejecutada por un robot, especificando la acción de movimiento considerando la dinámica del robot al desplazarse desde el espacio libre hacia una región de interés.

Finalmente, la planificación de un equipo robótico en entornos desconocidos es crucial (iv), especialmente en escenarios de búsqueda y rescate, donde los robots deben explorar el entorno de manera coordinada sin colisionar con obstáculos. Aquí se podrían abordar varias direcciones. Al comenzar a modelar el sistema robótico de forma probabilística, los robots podrían proporcionar información sobre el entorno basada en su distancia hacia las regiones de interés. Otra idea podría investigar el modelado de la incertidumbre a través de la misión que debe cumplirse, como el formalismo de Lógica Temporal Probabilística.

Para todos los modelos propuestos, la adaptabilidad y la robustez de los marcos también podrían mejorarse al expandirlos hacia sistemas distribuidos y aplicar algoritmos de programación matemática para reducir de manera eficiente el tiempo de resolución computacional.

References

[1] C. Mahulea, M. Kloetzer, and R. González, *Path Planning of Cooperative Mobile Robots Using Discrete Event Models*. John Wiley & Sons, 2020.

[2] C. Mahulea, M. Kloetzer, and J.-J. Lesage, "Multi-robot path planning with boolean specifications and collision avoidance," *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 101–108, 2020.

[3] D. Valera, L. Belmonte, F. Molina, and A. López, "Greenhouse agriculture in almería: A comprehensive techno-economic analysis," *Cajamar Caja Rural: Almería, Spain*, p. 408, 2016.

[4] M. Kloetzer and C. Mahulea, "Path planning for robotic teams based on LTL specifications and Petri net models," *Discrete Event Dynamic Systems*, vol. 30, no. 1, pp. 55–79, 2020.

[5] J. Gregory, J. Fink, E. Stump, J. Twigg, J. Rogers, D. Baran, N. Fung, and S. Young, "Application of multi-robot systems to disaster-relief scenarios with limited communication," in *Field and Service Robotics: Results of the 10th International Conference*, pp. 639–653, Springer, 2016.

[6] S. Tang, K. Sreenath, and V. Kumar, "Multi-robot trajectory generation for an aerial payload transport system," in *Robotics Research: The 18th International Symposium ISRR*, pp. 1055–1071, Springer, 2019.

[7] S. Liu, J. Shen, W. Tian, J. Lin, P. Li, and B. Li, "Balanced task allocation and collision-free scheduling of multi-robot systems in large spacecraft structure manufacturing," *Robotics and Autonomous Systems*, vol. 159, p. 104289, 2023.

[8] P. Garre and A. Harish, "Autonomous agricultural pesticide spraying uav," in *IOP Conference Series: Materials Science and Engineering*, vol. 455, p. 012030, IOP Publishing, 2018.

[9] D. C. Tsouros, A. Triantafyllou, S. Bibi, and P. G. Sarigannidis, "Data acquisition and analysis methods in uav-based applications for precision agriculture," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 377–384, IEEE, 2019.

[10] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *Ieee Access*, vol. 7, pp. 48572–48634, 2019.

[11] K. Noda and H. Aizawa, "Indoor environmental monitoring system using a robot vacuum cleaner," *Sensors and Materials*, vol. 32, no. 3, pp. 1133–1140, 2020.

[12] B. Mishra, D. Garg, P. Narang, and V. Mishra, "Drone-surveillance for search and rescue in natural disaster," *Computer Communications*, vol. 156, pp. 1–10, 2020.

[13] M. M. Abrar, R. Islam, and M. A. H. Shanto, "An autonomous delivery robot to prevent the spread of coronavirus in product delivery system," in *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0461–0466, IEEE, 2020.

[14] R. Monica, D. L. Rizzini, and S. Caselli, "Robot manipulation of tomato fruits using a commercial soft gripper," in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–7, IEEE, 2024.

[15] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[16] A. Benkrid, A. Benallegue, and N. Achour, "Multi-robot coordination for energy-efficient exploration," *Journal of Control, Automation and Electrical Systems*, vol. 30, no. 6, pp. 911–920, 2019.

[17] M. H. Cohen and C. Belta, "Model-based reinforcement learning for approximate optimal control with temporal logic specifications," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pp. 1–11, 2021.

[18] C. Mahulea and M. Kloetzer, "Robot planning based on boolean specifications using Petri net models," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2218–2225, 2017.

[19] J. Esparza, J. Křetínskỳ, and S. Sickert, "One theorem to rule them all: A unified translation of LTL into ω-automata," in *33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 384–393, 2018.

[20] E. Clarke, O. Grumberg, and K. Hamaguchi, "Another look at LTL model checking," in *Computer Aided Verification: 6th International Conference, CAV'94 Stanford, California, USA, June 21–23, 1994 Proceedings 6*, pp. 415–427, Springer, 1994.

[21] G. Holzmann, *The Spin Model Checker, Primer and Reference Manual*. Reading, Massachusetts: Addison-Wesley, 2004.

[22] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[23] I. Hustiu, M. Kloetzer, and C. Mahulea, "Distributed path planning of mobile robots with LTL specifications," in *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 60–65, IEEE, 2020.

[24] R. Ehlers and A. Khalimov, "Fully generalized reactivity (1) synthesis," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 83–102, Springer, 2024.

[25] B. Lacerda, D. Parker, and N. Hawes, "Optimal policy generation for partially satisfiable co-safe LTL specifications.," in *IJCAI*, vol. 15, pp. 1587–1593, Citeseer, 2015.

[26] E. M. M. Clarke, D. Peled, and O. Grumberg, *Model checking*. MIT Press, 1999.

[27] M. Reynolds, "An axiomatization of full computation tree logic," *The Journal of Symbolic Logic*, vol. 66, no. 3, pp. 1011–1057, 2001.

[28] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI communications*, vol. 29, no. 1, pp. 151–162, 2016.

[29] P. Thati and G. Roşu, "Monitoring algorithms for metric temporal logic specifications," *Electronic Notes in Theoretical Computer Science*, vol. 113, pp. 145–162, 2005.

[30] S. Hustiu, A.-I. Iancu, and F.-A. Brasoveanu, "Integration of MITL for cobots workflow in a manipulating application," in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 01–08, IEEE, 2024.

[31] N. Kamide and N. Yamamoto, "Inconsistency-tolerant hierarchical probabilistic computation tree logic and its application to model checking.," in *ICAART (2)*, pp. 490–499, 2021.

[32] C.-I. Vasile, D. Aksaray, and C. Belta, "Time window temporal logic," *Theoretical Computer Science*, vol. 691, pp. 27–54, 2017.

[33] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3451–3458, 2022.

[34] S. Karaman and E. Frazzoli, "Vehicle routing problem with metric temporal logic specifications," in *2008 47th IEEE conference on decision and control*, pp. 3953–3958, IEEE, 2008.

[35] S. Hustiu, C. Mahulea, M. Kloetzer, and J.-J. Lesage, "On multi-robot path planning based on Petri net models and LTL specifications," *IEEE Transactions on Automatic Control*, vol. 69, no. 9, pp. 6373–6380, 2024.

[36] R. Peterson, A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoglu, "Decentralized safe reactive planning under twtl specifications," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6599–6604, IEEE, 2020.

[37] L. Lindemann, J. Nowak, L. Schönbächler, M. Guo, J. Tumova, and D. V. Dimarogonas, "Coupled multi-robot systems under linear temporal logic and signal temporal logic tasks," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 858–865, 2019.

[38] F. S. Barbosa, L. Lindemann, D. V. Dimarogonas, and J. Tumova, "Integrated motion planning and control under metric interval temporal logic specifications," in *18th European Control Conference (ECC)*, pp. 2042–2049, 2019.

[39] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas, "Cooperative planning for coupled multi-agent systems under timed temporal specifications," in *2017 American Control Conference (ACC)*, pp. 1847–1852, 2017.

[40] M. Guo, J. Tumova, and D. V. Dimarogonas, "Cooperative decentralized multi-agent control under local LTL tasks and connectivity constraints," in *53rd IEEE conference on decision and control*, pp. 75–80, IEEE, 2014.

[41] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–31, 2019.

[42] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[43] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Decomposition of finite LTL specifications for efficient multi-agent planning," in *Distributed Autonomous Robotic Systems: The 13th International Symposium*, pp. 253–267, Springer, 2018.

[44] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[45] P. Yu and D. V. Dimarogonas, "Distributed motion coordination for multi-robot systems under LTL specifications," *arXiv preprint arXiv:2103.09111*, 2021.

[46] B. Lacerda and P. U. Lima, "Petri net based multi-robot task coordination from temporal logic specifications," *Robotics and Autonomous Systems*, vol. 122, pp. 343–352, 2019.

[47] C. Seatzu, M. Silva, and J. H. Van Schuppen, *Control of discrete-event systems*, vol. 433. Springer, 2013.

[48] M.-H. Matcovschi, C. Mahulea, and O. Pastravanu, "Petri net toolbox for matlab," in *11th IEEE Mediterranean Conference on Control and Automation MED'03*, 2003.

[49] W. J. Thong and M. Ameedeen, "A survey of Petri net tools," in *Advanced Computer and Communication Engineering Technology: Proceedings of the 1st International Conference on Communication and Computer Engineering*, pp. 537–551, Springer, 2015.

[50] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta, "Automatic deployment of robotic teams," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 75–86, 2011.

[51] E. Montijano and C. Mahulea, "Probabilistic Multi-Robot Path Planning with High-Level Specifications using Petri Net Models," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 2188–2193, IEEE, 2021.

[52] H. Costelha and P. Lima, "Robot task plan representation by Petri nets: modelling, identification, analysis and execution," *Autonomous Robots*, vol. 33, no. 4, pp. 337–360, 2012.

[53] J. Ezpeleta and J. M. Colom, "Automatic synthesis of colored Petri nets for the control of fms," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 327–337, 1997.

[54] Z. He, Y. Dong, G. Ren, C. Gu, and Z. Li, "Path planning for automated guided vehicle systems with time constraints using timed Petri nets," *Measurement and Control*, vol. 53, no. 9-10, pp. 2030–2040, 2020.

[55] T. Le Moigne, C. Mahulea, and G. Faraut, "Optimizing clinical pathways: A probabilistic time Petri net approach," *IFAC-PapersOnLine*, vol. 58, no. 1, pp. 96–101, 2024.

[56] Z. He, R. Zhang, N. Ran, and C. Gu, "Path planning of multi-type robot systems with time windows based on timed colored Petri nets," *Applied Sciences*, vol. 12, no. 14, p. 6878, 2022.

[57] K. Jensen and G. Rozenberg, *High-level Petri nets: theory and application*. Springer Science & Business Media, 2012.

[58] R. Valk, "Object Petri nets," in *Advanced Course on Petri Nets*, pp. 819–848, Springer, 2003.

[59] S. Hustiu, I. Hustiu, M. Kloetzer, and C. Mahulea, "LTL task decomposition for 3D high-level path planning," *Journal of Control Engineering and Applied Informatics*, vol. 23, no. 3, pp. 76–87, 2021.

[60] S. Hustiu, D. V. Dimarogonas, C. Mahulea, and M. Kloetzer, "Multi-robot motion planning under MITL specifications based on time Petri nets," in *2023 European Control Conference (ECC)*, pp. 1–8, IEEE, 2023.

[61] S. Hustiu, E. Robillard, J. Ezpeleta, C. Mahulea, and M. Kloetzer, "Multi-robot motion planning based on nets-within-nets modeling and simulation," *arXiv preprint arXiv:2304.08772*, 2023.

[62] C. E. Leiserson, R. L. Rivest, T. H. Cormen, and C. Stein, *Introduction to algorithms*, vol. 3. MIT press, 1994.

[63] A. R. Khairuddin, M. S. Talib, and H. Haron, "Review on simultaneous localization and mapping (slam)," in *2015 IEEE international conference on control system, computing and engineering (ICCSCE)*, pp. 85–90, IEEE, 2015.

[64] M. Kallmann and M. Kapadia, "Geometric and discrete path planning for interactive virtual worlds," in *ACM SIGGRAPH 2016 Courses*, pp. 1–29, 2016.

[65] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, pp. 247–253, 2000.

[66] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, pp. 1–9, 1974.

[67] R. Shewchuk, "Star splaying: an algorithm for repairing delaunay triangulations and convex hulls," in *Proceedings of the twenty-first annual symposium on Computational geometry*, pp. 237–246, 2005.

[68] M. Lupascu, S. Hustiu, A. Burlacu, and M. Kloetzer, "Path planning for autonomous drones using 3d rectangular cuboid decomposition," in *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 119–124, IEEE, 2019.

[69] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[70] S. Hustiu, M. Kloetzer, and A. Burlacu, "Collision Free Path Planning for Unmanned Aerial Vehicles in Environments with Dynamic Obstacles," in *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 520–525, IEEE, 2020.

[71] B. Grünbaum, *Convex Polytopes*. Springer-Verlag, 2003.

[72] R. Alur and D. Dill, "The theory of timed automata," in *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, pp. 45–73, Springer, 1991.

[73] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2008.

[74] M. Zhou and F. DiCesare, *Petri net synthesis for discrete event control of manufacturing systems*, vol. 204. Springer Science & Business Media, 2012.

[75] M. Kloetzer, C. Mahulea, and J.-M. Colom, "Petri net approach for deadlock prevention in robot planning," in *IEEE 18th Conf. on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–4, 2013.

[76] J. L. Peterson, "Petri nets," *ACM Computing Surveys (CSUR)*, vol. 9, no. 3, pp. 223–252, 1977.

[77] L. C. G. J. M. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Transactions on Automatic Control*, vol. 51, pp. 938–948, 2006.

[78] C. Belta and L. Habets, "Controlling a class of nonlinear systems on rectangles," *IEEE Transactions on Automatic Control*, vol. 51, no. 11, pp. 1749–1759, 2006.

[79] P. Merlin and D. Farber, "Recoverability of communication protocols-implications of a theoretical study," *IEEE transactions on Communications*, vol. 24, no. 9, pp. 1036–1043, 1976.

[80] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets," *IEEE transactions on software engineering*, vol. 17, no. 3, p. 259, 1991.

[81] L. Popova-Zeugmann and L. Popova-Zeugmann, *Time Petri nets*. Springer, 2013.

[82] L. Popova-Zeugmann, "Time Petri nets: Theory, tools and applications part i," 2008. ATPN 2008, Xi'an, China.

[83] J. E. Coolahan and N. Roussopoulos, "Timing requirements for time-driven systems using augmented Petri nets," *IEEE transactions on software engineering*, no. 5, pp. 603–616, 1983.

[84] J. Wang, *Timed Petri nets: Theory and application*, vol. 9. Springer Science & Business Media, 2012.

[85] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017.

[86] J. King, R. K. Pretty, and R. G. Gosine, "Coordinated execution of tasks in a multiagent environment," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 33, no. 5, pp. 615–619, 2003.

[87] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[88] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *2008 47th IEEE Conference on Decision and Control*, pp. 2117–2122, IEEE, 2008.

[89] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. Pappas, "Symbolic planning and control of robot motion," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 61–71, 2007.

[90] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[91] F. Blahoudek, *Translation of an LTL fragment to deterministic Rabin and Streett automata*. PhD thesis, Master's thesis, Masarykova Univerzita, 2012.

[92] P. Wolper, M. Vardi, and A. Sistla, "Reasoning about infinite computation paths," in *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science* (E. N. et al., ed.), (Tucson, AZ), pp. 185–194, 1983.

[93] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Proceedings of the 13th Conference on Computer Aided Verification (CAV'01)* (H. C. G. Berry and A. Finkel, eds.), no. 2102 in LNCS, pp. 53–65, SPRINGER, 2001.

[94] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu, "Spot 2.0 - a framework for LTL and ω-automata manipulation," in *In: Proc. of ATVA'16*, no. 9938 in LNCS, pp. 122–129, 2016.

[95] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *Journal of the ACM (JACM)*, vol. 43, no. 1, pp. 116–146, 1996.

[96] O. Maler, D. Nickovic, and A. Pnueli, "From MITL to timed automata," in *International conference on formal modeling and analysis of timed systems*, pp. 274–289, Springer, 2006.

[97] D. Ničković and N. Piterman, "From MTL to deterministic timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 152–167, Springer, 2010.

[98] S. Andersson, "Automatic control design synthesis under metric interval temporal logic specifications," 2016.

[99] D. D'Souza and P. Prabhakar, "On the expressiveness of MTL in the pointwise and continuous semantics," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 1, pp. 1–4, 2007.

[100] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O. H. Roux, "Comparison of the expressiveness of timed automata and time Petri nets," in *International conference on formal modeling and analysis of timed systems*, pp. 211–225, Springer, 2005.

[101] R. Alur, "Timed automata," in *International Conference on Computer Aided Verification*, pp. 8–22, Springer, 1999.

[102] G. E. Fainekos and G. J. Pappas, "Robust sampling for MITL specifications," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 147–162, Springer, 2007.

[103] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pp. 152–166, Springer, 2004.

[104] O. Maler, D. Nickovic, and A. Pnueli, "Checking temporal properties of discrete, timed and continuous behaviors," *Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, pp. 475–505, 2008.

[105] I. Hustiu, M. Kloetzer, and C. Mahulea, "Extension of a decomposition method for a global LTL specification," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–4, IEEE, 2023.

[106] J. Tumova and D. V. Dimarogonas, "Decomposition of multi-agent planning under distributed motion and task LTL specifications," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 7448–7453, IEEE, 2015.

[107] M. Kloetzer and C. Mahulea, "Path planning for robotic teams based on LTL specifications and Petri net models," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 30, no. 1, pp. 55–79, 2020.

[108] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[109] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, Massachusetts and New York: The MIT Press and McGraw-Hill Book Company, 2nd ed., 2001.

[110] IBM, "IBM ILOG CPLEX Optimization Studio. Software." http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/, 2016.

[111] S. Reveliotis and E. Roszkowska, "Conflict Resolution in Free-Ranging Multivehicle Systems: A Resource Allocation Paradigm," *IEEE Trans. on Robotics*, vol. 27, no. 2, pp. 283–296, 2011.

[112] H. Sofia, H. Ioana, K. Marius, and M. Cristian, "LTL task decomposition for 3d high-level path planning in known and static environments." [Available at] https://youtu.be/awzeIZbexng.

[113] B. Lacerda and P. U. Lima, "Petri net based multi-robot task coordination from temporal logic specifications," *Robotics and Autonomous Systems*, vol. 122, p. 103289, 2019.

[114] E. Vitolo, C. Mahulea, and M. Kloetzer, "A computationally efficient solution for path planning of mobile robots with boolean specifications," in *ICSTCC'2017: 21st International Conference on System Theory, Control and Computing*, (Sinaia, Romania), pp. 63–69, 2017.

[115] K. Etessami, "Stutter-invariant languages, ω-automata, and temporal logic," in *International Conference on Computer Aided Verification*, pp. 236–248, Springer, 1999.

[116] S. F. Roselli, P.-L. Götvall, M. Fabian, and K. Åkesson, "A compositional algorithm for the conflict-free electric vehicle routing problem," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1405–1421, 2022.

[117] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear programming and network flows*. John Wiley & Sons, 2011.

[118] L. Parrilla, C. Mahulea, and M. Kloetzer, "Rmtool: recent enhancements," in *IFAC-PapersOnLine*, no. 1, pp. 5824–5830, 2017.

[119] R. González, C. Mahulea, and M. Kloetzer, "A Matlab-based interactive simulator for mobile robotics," in *IEEE CASE'2015: Int. Conf. on Autom. Science and Engineering*, 2015.

[120] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative robots and sensor networks 2015*, pp. 31–51, 2015.

[121] L. Antonyshyn, J. Silveira, S. Givigi, and J. Marshall, "Multiple mobile robot task and motion planning: A survey," *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–35, 2023.

[122] J. Tang, X. Chen, X. Zhu, and F. Zhu, "Dynamic reallocation model of multiple unmanned aerial vehicle tasks in emergent adjustment scenarios," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 2, pp. 1139–1155, 2022.

[123] F. Faruq, B. Lacerda, N. Hawes, and D. Parker, "A framework for simultaneous task allocation and planning under uncertainty," *ACM Transactions on Autonomous and Adaptive Systems*, 2024.

[124] M. Lawley, S. Reveliotis, and P. Ferreira, "The application and evaluation of banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems," *International Journal of Flexible Manufacturing Systems*, vol. 10, no. 1, pp. 73–100, 1998.

[125] L. Kalinovcic, T. Petrovic, S. Bogdan, and V. Bobanac, "Modified Banker's algorithm for scheduling in multi-AGV systems," in *International Conference on Automation Science and Engineering*, pp. 351–356, IEEE, 2011.

[126] V. Bobanac and S. Bogdan, "Routing and scheduling in multi-AGV systems based on dynamic banker algorithm," in *16th Mediterranean Conference on Control and Automation*, pp. 1168–1173, IEEE, 2008.

[127] D. Song, Y. Li, and T. Song, "Modified Banker's algorithm with dynamically release resources," in *International Conference on Communications, Information System and Computer Engineering*, pp. 566–569, IEEE, 2021.

[128] F. Tricas, J. M. Colom, and J. Ezpeleta, "Some improvements to the Banker's algorithm based on the process structure," in *International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 3, pp. 2853–2858, IEEE, 2000.

[129] D. Yu, X. Hu, K. Liang, and J. Ying, "A parallel algorithm for multi-AGV systems," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–15, 2021.

[130] H. Sofia, M. Cristian, and K. Marius, "Parallel motion execution and pathrerouting for a team of mobile robots." [Available at] https://youtu.be/F2UvUzFydpY.

[131] B. Berthomieu, F. Peres, and F. Vernadat, "Bridging the gap between timed automata and bounded time Petri nets," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 82–97, Springer, 2006.

[132] Z. Xu and A. A. Julius, "Census signal temporal logic inference for multiagent group behavior analysis," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 264–277, 2016.

[133] A. Giua, F. DiCesare, and M. Silva, "Generalized mutual exclusion constraints for Petri nets with uncontrollable transitions," in *IEEE Int. Conf. on Systems, man, and cybernetics*, pp. 947–949, 1992.

[134] G. Gardey, D. Lime, M. Magnin, and O. Roux, "Romeo: A tool for analyzing time Petri nets," in *International Conference on Computer Aided Verification*, pp. 418–423, Springer, 2005.

[135] M. Momeni, J. Relefors, A. Khatry, L. Pettersson, A. V. Papadopoulos, and T. Nolte, "Automated fabrication of reinforcement cages using a robotized production cell," *Automation in Construction*, vol. 133, p. 103990, 2022.

[136] O. Kummer, F. Wienberg, M. Duvigneau, M. Köhler, D. Moldt, and H. Rölke, "Renew–the reference net workshop," in *Tool Demonstrations, 21st International Conference on Application and Theory of Petri Nets, Computer Science Department, Aarhus University, Aarhus, Denmark*, pp. 87–89, 2000.

[137] "Nets-within-Nets for Motion Planning with Renew." https://eva-robillard.github.io/, 2024. Accessed: 24-11-2024.

[138] C. Ju and H. I. Son, "A hybrid systems-based hierarchical control architecture for heterogeneous field robot teams," *IEEE Transactions on Cybernetics*, 2021.

[139] J. J. Roldán, P. Garcia-Aunon, M. Garzón, J. De León, J. Del Cerro, and A. Barrientos, "Heterogeneous multi-robot system for mapping environmental variables of greenhouses," *Sensors*, vol. 16, no. 7, 2016.

[140] M. Allison and M. Spradling, "Modeling sub-team formations for heterogeneous multi-robot systems using colored Petri-net semantics," in *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 237–243, IEEE, 2022.

[141] M. Köhler, D. Moldt, and H. Rölke, "Modelling mobility and mobile agents using Mets within nets," in *ICATPN 2003: Applications and Theory of Petri Nets, Eindhoven, The Netherlands, June 23–27*, pp. 121–139, Springer, 2003.

[142] P. Alvarez, J. A. Banares, and J. Ezpeleta, "Approaching web service coordination and composition by means of Petri Nets. the case of the nets-within-nets paradigm," in *International Conference on Service-Oriented Computing*, pp. 185–197, Springer, 2005.

[143] Y. Kissoum, R. Maamri, and Z. Sahnoun, "Modeling smart home using the paradigm of nets within nets," in *AIMSA 2012: Artificial Intelligence: Methodology, Systems, and Applications, Varna, Bulgaria, September 12-15, 2012.*, pp. 286–295, Springer, 2012.

[144] R. Bardini, A. Benso, S. Di Carlo, G. Politano, and A. Savino, "Using Nets-within-nets for modeling differentiating cells in the epigenetic landscape," in *IWBBIO 2016: Bioinformatics and Biomedical Engineering, Granada, Spain, April 20-22, 2016*, pp. 315–321, Springer, 2016.

[145] R. Bardini, G. Politano, A. Benso, and S. Di Carlo, "Using multi-level Petri nets models to simulate microbiota resistance to antibiotics," in *BIBM 2017: Int. Conf. on Bioinf. and Biomedicine*, pp. 128–133, 2017.

[146] S. Willrodt, D. Moldt, and M. Simon, "Modular model checking of reference nets: Momoc.," in *PNSE@ Petri Nets*, pp. 181–193, 2020.

[147] L. Capra and M. Köhler-Bußmeier, "Encoding Nets-Within-Nets in maude," in *Science and Information Conference*, pp. 355–372, Springer, 2023.

[148] M. Figat and C. Zieliński, "Methodology of designing multi-agent robot control systems utilising hierarchical Petri nets," in *ICRA 2019: Int. Conf. on Robotics and Automation*, pp. 3363–3369, 2019.

[149] M. P. Cabasino, A. Giua, M. Pocci, and C. Seatzu, "Discrete event diagnosis using labeled Petri nets. an application to manufacturing systems," *Control Engineering Practice*, vol. 19, no. 9, pp. 989–1001, 2011.

[150] K. Jensen, "Coloured Petri nets: A high level language for system design and analysis," in *High-level Petri Nets*, pp. 44–119, Berlin, Heidelberg: Springer Berlin Heidelberg, 1991.

[151] J. Desel and W. Reisig, "Place/transition Petri nets," in *Advanced Course on Petri Nets*, pp. 122–173, Springer, 1996.

[152] MATLAB, "Optimization toolbox."

[153] L. Cabac, M. Haustermann, and D. Mosteller, "Renew 2.5–towards a comprehensive integrated development environment for Petri net-based applications," in *Application and Theory of Petri Nets and Concurrency: 37th International Conference, Petri NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings 37*, pp. 101–112, Springer, 2016.

[154] C. Mahulea, "Robotmotiontoolbox-rmtool-under-matlab." https://github.com/cmahulea/RobotMotionToolbox-RMTool-under-MATLAB, 2024. Available at: https://github.com/cmahulea/RobotMotionToolbox-RMTool-under-MATLAB.

[155] The MathWorks, Inc., "Matlab optimization toolbox." Accessed: 2024-11-10, 2024. https://www.mathworks.com/products/optimization.html.

[156] A. Makhorin, "GNU linear programming kit." http://www.gnu.org/software/glpk/, 2012.

[157] "Simple example for nets-within-nets - github implementation." https://github.com/eva-robillard/NWN_Simple. Accessed: 24-11-2024.

[158] "Complex example for nets-within-nets - github implementation." https://github.com/eva-robillard/NWN_Complex. Accessed: 24-11-2024.

[159] R. Janet, W. Richard, S. Tim, and H. Craing, "World Resources Institute." https://www.wri.org/insights/how-sustainably-feed-10-billion-people-2050-21-charts, 2018. [Online; accessed 12-June-2022].

[160] J. Holland, L. Kingston, C. McCarthy, E. Armstrong, P. O'Dwyer, F. Merz, and M. McConnell, "Service robots in the healthcare sector," *Robotics*, vol. 10, no. 1, p. 47, 2021.

[161] A. López-Martínez, D. L. Valera-Martínez, F. D. Molina-Aiz, M. d. l. Á. Moreno-Teruel, A. Peña-Fernández, and K. E. Espinoza-Ramos, "Analysis of the effect of concentrations of four whitening products in cover transmissivity of mediterranean greenhouses," *International Journal of Environmental Research and Public Health*, vol. 16, no. 6, p. 958, 2019.

[162] J. Gázquez, J. López, J. Pérez-Parra, E. Baeza, P. Lorenzo, I. Caparros, *et al.*, "Effects of three cooling systems on the microclimate of a greenhouse with a pepper crop in the mediterranean area," *Acta Horticulturae*, vol. 927, pp. 739–746, 2012.

[163] J. Pérez-Alonso, Á. Carreño-Ortega, F. J. Vázquez-Cabrera, and Á. J. Callejón-Ferre, "Accidents in the greenhouse-construction industry of se spain," *Applied Ergonomics*, vol. 43, no. 1, pp. 69–80, 2012.

[164] F. D. M. Aiz, *Simulación y modelación de la ventilación en invernaderos de Almería mediante la utilización de dinámica computacional de fluidos*. PhD thesis, Universidad de Almería, 2010.

[165] D. Valera, L. Belmonte, F. Molina-Aiz, A. López, and F. Camacho, "The greenhouses of almería, spain: Technological analysis and profitability," in *International Symposium on New Technologies and Management for Greenhouses-GreenSys2015 1170*, pp. 219–226, 2015.

[166] S. Hustiu, M. Kloetzer, and C. Mahulea, "Mission assignment and 3d path planning for a team of UAVs," in *2021 25th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 401–406, IEEE, 2021.

[167] U. Pferschy and R. Staněk, "Generating subtour elimination constraints for the TSP from pure integer solutions," *Central European Journal of Operations Research*, vol. 25, pp. 231–260, 2017.

[168] S. Ghani, F. Bakochristou, E. M. A. A. ElBialy, S. M. A. Gamaledin, M. M. Rashwan, A. M. Abdelhalim, and S. M. Ismail, "Design challenges of agricultural greenhouses in hot and arid environments–a review," *Engineering in Agriculture, Environment and Food*, vol. 12, no. 1, pp. 48–70, 2019.

[169] K. G. Hollingsworth, "Reducing acquisition time in clinical MRI by data undersampling and compressed sensing reconstruction," *Physics in Medicine & Biology*, vol. 60, no. 21, p. R297, 2015.

[170] C. Hennersperger, B. Fuerst, S. Virga, O. Zettinig, B. Frisch, T. Neff, and N. Navab, "Towards mri-based autonomous robotic us acquisitions: a first feasibility study," *IEEE transactions on medical imaging*, vol. 36, no. 2, pp. 538–548, 2016.

[171] K. Li, Y. Xu, and M. Q.-H. Meng, "An overview of systems and techniques for autonomous robotic ultrasound acquisitions," *IEEE Trans. on Medical Robotics and Bionics*, vol. 3, no. 2, pp. 510–524, 2021.

[172] M. Kloetzer and C. Mahulea, "Path planning for robotic teams based on LTL specifications and Petri net models," *Discrete Event Dyn. Syst.: Theory Appl.*, vol. 30, pp. 55–79, Mar. 2020.

[173] I. I. Cplex, "V12. 1: User's manual for CPLEX," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[174] L. Busoniu, B. De Schutter, and R. Babuska, "Decentralized reinforcement learning control of a robotic manipulator," in *2006 9th International Conference on Control, Automation, Robotics and Vision*, pp. 1–6, IEEE, 2006.

[175] D. Panescu and C. Pascal, "Holonic coordination obtained by joining the contract net protocol with constraint satisfaction," *Computers in Industry*, vol. 81, pp. 36–46, 2016.

[176] X. F. Zha, "Optimal pose trajectory planning for robot manipulators," *Mechanism and Machine Theory*, vol. 37, no. 10, pp. 1063–1086, 2002.

[177] H. Deng, L. Zhu, J. Wang, M. Chen, H. Wu, and C. He, "Kinematics modeling and trajectory planning of KUKA manipulator based on MATLAB," in *Journal of Physics: Conference Series*, vol. 2216, p. 012056, IOP Publishing, 2022.

[178] M. Guertler, L. Tomidei, N. Sick, M. Carmichael, G. Paul, A. Wambsganss, V. H. Moreno, and S. Hussain, "When is a robot a cobot? moving beyond manufacturing and arm-based cobot manipulators," *Proceedings of the Design Society*, vol. 3, pp. 3889–3898, 2023.

[179] I. Sopon, C. Tincu, A. Ganciu, and A. Burlacu, "Mixed reality framework for eye-in-hand collaborative robot-human interaction," in *2023 27th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 303–308, IEEE, 2023.

[180] Z. Feng, G. Hu, Y. Sun, and J. Soon, "An overview of collaborative robotic manipulation in multi-robot systems," *Annual Reviews in Control*, vol. 49, pp. 113–127, 2020.

[181] L. Liu, F. Guo, Z. Zou, and V. G. Duffy, "Application, development and future opportunities of collaborative robots (cobots) in manufacturing: A literature review," *International Journal of Human–Computer Interaction*, vol. 40, no. 4, pp. 915–932, 2024.

[182] S. Parascho, I. X. Han, S. Walker, A. Beghini, E. P. Bruun, and S. Adriaenssens, "Robotic vault: a cooperative robotic assembly method for brick vault construction," *Construction Robotics*, vol. 4, pp. 117–126, 2020.

[183] K. Zbiss, A. Kacem, M. Santillo, and A. Mohammadi, "Automatic collision-free trajectory generation for collaborative robotic car-painting," *IEEE Access*, vol. 10, pp. 9950–9959, 2022.

[184] J. Vannoy and J. Xiao, "Real-time tight coordination of mobile manipulators in unknown dynamic environments," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2513–2519, IEEE, 2007.

[185] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Accessed: 2024-04-28].

[186] H. Sofia, I. Andrei-Iulian, and B. Florian-Alexandru, "Integration of MITL for cobots workflow in manipulating application." [Available at] https://youtu.be/4IQ0_5Uv9bQ.

[187] U. Robots, "The UR5 cobot." https://www.universal-robots.com/media/1828033/ur5_tech_spec_web_en.pdf, 2018. [Accessed: 2024-04-28].

[188] L. Robotics, "The LM3 robot." https://lebai.ltd/, 2017. [Accessed: 2024-04-28].