RMTool: Recent Enhancements [⋆]

Luis Parrilla [*] **Cristian Mahulea** [*] **Marius Kloetzer** [**]

[*] *Aragón Institute of Engineering Research (I3A), University of Zaragoza, María de luna 1, 50018 Zaragoza, Spain (e-mails: {parrilla,cmahulea}@unizar.es)*
[**] *Department of Automatic Control and Applied Informatics, Technical University of Iasi, Iasi 700050, Romania (e-mail: kmarius@ac.tuiasi.ro)*

Abstract:
This paper presents the enhancements introduced in the second version of RMTool, an open-source Matlab-based interactive software for teaching mobile robotics in introductory courses. In the first version only navigation problem of one robot has been considered (the robot should avoid obstacles and reach a desired position). In the actual version, the toolbox introduces new algorithms to cope with modeling and path planning of multiple identical robots, where the final states of the robots and the regions visited along trajectories should satisfy Linear Temporal Logic (LTL) or Boolean-based formulas. The paper includes extensive simulation results performed in RMTool, pertained to multi-robot path planning based on an LTL or Boolean team specification.

Keywords: Software Tool, Discrete Event Systems, Mobile Robots.

1. INTRODUCTION

Matlab is a numerical computing environment widely used in industry by engineers and scientists, as well as in education. Many mobile robotics simulator toolboxes had been developed in Matlab. A precursor work of this field, simulation industrial robots, was (Corke, 1996). Others like SIMROBOT (Hrabeck, 2001) or MRSim (Couceiro, 2012), an extension from the first one, allow to simulate the behavior of one or more mobile robots that could be equipped with several virtual sensors. A MATLAB toolbox containing various routines related to planning and trajectory following for mobile robots is described in (Corke, 2011).

RMTool (Gonzalez et al., 2015) is an open-source Matlab-based interactive software tool for teaching mobile robotics in introductory courses, focused on the following concepts: (a) robot modeling, (b) path planning, and (c) motion control, rather than on the program implementation (the users do not need to have any previous knowledge on Matlab or programming).

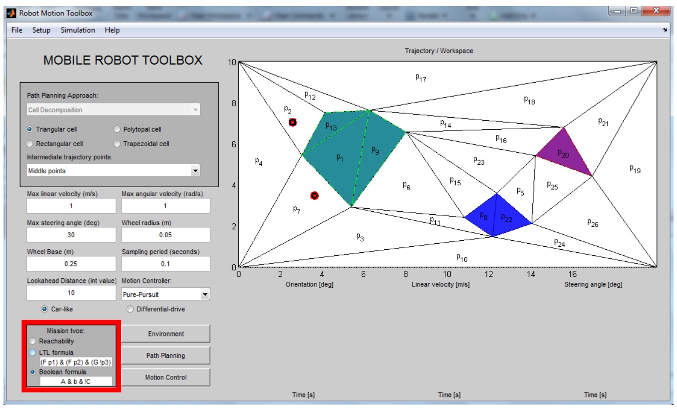
Two of the most popular robot kinematic models are considered in order to simulate the robot's motion (Dudek and Jenkin, 2010): (i) car-like robot and (ii) differential-drive robot. (i) The first one is a Four-wheeled car-like robot modeled with fixed rear wheels and rotatory (about the vertical axis) front wheels to steer the vehicle. (ii) The second one uses two parallel driving wheels mounted on an axis and combines different speed and spin direction for each wheel to drive the vehicle in the desired way. Both models can be configured with respect to settings as the robot radius, velocity and distance between the wheels.

RMTool includes different methods for mobile robots' path planning. The path planning problems solved in the first version, refers to finding a path (reference trajectory) from an initial point to an end point in a known 2D environment cluttered with convex polygonal obstacles and one robot. Two general strategies were considered: (i) cell decompositions and (ii) roadmaps. The technique of *cell decomposition* divides the free space into a set of regions having the same geometrical shape (De Berg et al., 2000)(i.e., triangular, trapezoidal, polytopal, or rectangular shape). From this partition a finite graph is constructed and running the Dijkstra algorithm on it, we obtain the sequence of cells to be followed by the robot from the initial to the desired position while avoiding the obstacles.

The *roadmap* strategy identifies a set of fast routes within the free space. Two methods are developed using this strategy: (a) visibility graph and (b) generalized Voronoi diagram. The *visibility graph* approach (V-graph) solves a minimum-length path by following a sequence of edges connecting a set of obstacle vertices visible one from the other, as well as the initial and end points (Siegwart et al., 2011), (Choset, 2005). *Generalized Voronoi diagram* in contraposition tends to maximize the distance between the robot and obstacles in the map (Dudek and Jenkin, 2010). For polygonal obstacles this diagram contains only equidistant points from two or more closest obstacles and this leads to straight and parabolic segments.

As it can be seen in Fig. 1, the toolbox presents a friendly GUI that enables the user to easily insert the input parameters in order to perform the simulations,

[⋆] This work has been partially supported by CICYT - FEDER project DPI2014-57252-R.


Robot Motion Toolbox

File Setup Simulation Help

MOBILE ROBOT TOOLBOX

Path Planning Approach: Cell Decomposition

[✓ Triangular cell] Polytopal cell Rectangular cell Trapezoidal cell

Intermediate trajectory points: Middle points

Max linear velocity (m/s): 1	Max angular velocity (rad/s): 1
Max steering angle (deg): 30	Wheel radius (m): 0.05
Wheel Base (m): 0.25	Sampling period (seconds): 0.1
Lookahead Distance (int value): 10	Motion Controller: Pure-Pursuit

[✓ Car-like] Differential-drive

Mission type: Reachability; LTL formula: (F p1) & (F p2) & (G !p3); [✓ Boolean formula]: A & b & !C

Fig. 1. The main window of RMTool. The red box indicates the user input for new functionalities presented in this work.

and it displays the results in a graphical environment. In this figure is shown the new menu to select the added functionalities available in the tool.

The recent enhancements reported by this work implement new algorithms for the path planning problem, where a team of identical robots should perform some motion tasks. The environment contains a set of known and fixed regions of interest, while the team mission belongs to classes of specifications on visiting or avoiding the regions of interest. Two algorithmic planning methods are introduced to RMTool, each for a task specification formalism: LTL (Linear Temporal Logic) missions by using transition system models (described in Sec. 2), and boolean-based missions by using Petri net models (described in Sec. 3).

2. LTL MISSIONS AND TRANSITION SYSTEM MODELS

In this new version of the toolbox, a team of n identical and omnidirectional point robots is assumed to move in a rectangular environment. This environment contains a set Π of regions of interest and the robots are deployed in it. The environment is partitioned in a finite set of disjoint cells, by using in this case a cell decomposition approach (LaValle, 2006; Kloetzer and Mahulea, 2014) using triangular shapes. After this decomposition, each region of interest π_i corresponds to one or more cells of the environment, and if one or more robots are located in any of these cells the proposition π_i is considered satisfied (it is *true*).

Here, instead of giving end position for each robot, the motion task for the entire team is given by a Linear Temporal Logic (LTL) formula (Baier and Katoen, 2008). LTL is often encountered as a formalism for expressing high level robotic tasks (Fainekos et al., 2009; Ding et al., 2011; Guo et al., 2014). This toolbox considers LTL_{-X} formulas, that unlike standard LTL do not use the "next" operator, which is meaningless for continuous trajectories. The set of propositions $\Pi = \{\Pi_1, \Pi_2, ..., \Pi_{|\Pi|}\}$ is used to give an LTL_{-X} formula, φ, defining the task that the team of robots should fulfill. An LTL_{-X} formula is recursively defined over the set of atomic propositions Π, by using standard boolean operators (RMTool symbols): negation (!), disjunction (|), conjunction (&), implication (->) and equivalence (<->); and temporal operators with straightforward meaning: until (U), eventually (F) and always (G). Due to the logical and temporal operators, an LTL_{-X} task can indicate requirements as avoidance of some regions until others are visited, subsets of regions to be visited in a specific order, surveillance (infinite number of revisits) of some regions etc. Examples on such mission tasks can be found in the works referred in this paragraph, as well as in Ex. 1 and Sec. 4.

The main problem that we consider in this section is to plan the trajectory for each team member, such that the overall team motion satisfies the imposed LTL_{-X} task. Various formal methods were proposed for this problem for different assumptions, e.g., regarding knowledge on environment, heterogenous team, communication among robots. The purpose of this work is to integrate such methods in RMTool. We assume identical robots that can communicate with each other, scenario for which works as (Ding et al., 2011; Kloetzer and Mahulea, 2015) provide solutions based on transition system models. This work integrates these solutions into RMTool, and, for the presentation completeness, we briefly present in the following the main operations and ideas behind the involved formal methods.

Any LTL_{-X} formula over set Π can be transformed into a nondeterministic Büchi automaton that accepts all and only the input strings satisfying the formula (Wolper et al., 1983; Gastin and Oddoux, 2001). A Büchi automaton that corresponding to an LTL_{-X} formula over Π is defined as $B = (S, S_0, \Sigma_B, \rightarrow_B, F)$, where:

- S is a finite set of states;
- S_0 is the initial state;
- $\Sigma_B = 2^{\Pi}$ is the set of inputs;
- $\rightarrow_B$ is the set of transitions enabled by inputs;
- F is the final state.

The movement of a mobile robot can be abstracted into a finite state transition system T defined as a tuple $T = (Q, q_0, \delta_T, \Pi^T, \gamma)$ (Kloetzer and Mahulea, 2015),(Belta and Habets, 2006), where:

- Q is the finite number of states (cells of the environment's partition);
- q_0 is the initial state (cell) of the robot;
- $\delta_T \subseteq Q \times Q$ is the transition relation, with $(q, q') \in \delta_T$ if q and q' are neighboring cells and the robot can move from q to q' without visiting any other cell;
- $\Pi^T = \Pi$ is the set of regions of interest (possibly overlapping) that can be visited by the robot;
- $\gamma : Q \rightarrow 2^{\Pi^T}$ is the satisfaction map, where $\gamma(q)$ is the set of regions of Π that contain cell labeled by q.

From the transition system T of each robot (the transition system is the same for all the robots with different initial state on each one) we obtain a new transition system T_T that models the movement capabilities of the whole team of n robots, being $T_T = (Q_T, q_{0T}, \delta_T, O_T, \gamma_T, \omega_T^m)$ (Kloetzer and Mahulea, 2016):

- $Q_T = Q^n$ the cartesian product of Q n-times;
- $q_{0T} = (q_01, q_02, ...q_0n) \in Q_T$ the initial state;
- $\delta_T \subseteq Q_T \times Q_T$ is the transition relation, defined based on synchronized individual transitions δ_T from robot models T;

- $O^T = 2^\Pi$ is the set of possible observations;
- $\gamma_T : Q_T \to O^T$ is the satisfaction map, where $\gamma_T(q_1, q_2, .., q_n) = \cup_{i=1}^{n} \gamma(q_i)$;
- $\omega_T^m : \delta_T \to \mathbb{N}$ is a weighting function yielding the number of robots that move during a given transition.

In the transition system T_T may be multiple runs (sequences of transitions that satisfy the $LTL_{\text{-}X}$ formula). To find the solution that minimizes the total number of movements, a special product automaton $A = T_T \times B$ is built, namely $A = (S_A, S_{A0}, \delta_A, \omega_A^m, F_A)$, where:

- $S_A = Q_T \times S$ is the set of states;
- $S_{A0} = q_{0T} \times S_0$ is the set of initial states;
- $\delta_A \subseteq S_A \times S_A$ is the transition relation, based on formal links between team satisfaction map γ_T and transitions of Büchi automaton B;
- $\omega_A^m : \delta_A \to \mathbb{N}$ is a movement-based weighting function for transitions of A, inherited from ω_T^m;
- $F_A = Q_T \times F$ is the set of final states.

The automaton A extends the one from (Kloetzer and Mahulea, 2012) adding the number of moving robots for each transition of T_T in map ω_A^m.

The automaton A is used to find a run that satisfies the proposed condition by the $LTL_{\text{-}X}$ formula. The graph search used to find the solution uses an algorithm described in (Kloetzer and Mahulea, 2015). The algorithm finds in a single run the optimum paths from an initial node to all final nodes, and then it finds an accepted run of A by iterating at most $|S_{A0}| + |F_A|$ searches. The accepted run of A is projected to individual robot trajectories (runs of models T). These are infinite runs having a so-called prefix-suffix structure. Due to definition of T_T, the robots have to synchronize with each other when moving between adjacent partition cells. As detailed in (Kloetzer and Mahulea, 2015), the overall planning method from this section has a high complexity, mainly due to the number of states of automaton A, $|A| = |B| \times |T|^n$. Note that the size of A is drastically affected by the number of partition cells and the number of robots, while size of B can be in worst cases double exponential in the size of $LTL_{\text{-}X}$ formula (Gastin and Oddoux, 2001).

Example 1. Lets consider an environment with 5 regions of interest and two robots deployed as in Fig. 2. The team of robots has to move such that it satisfies the $LTL_{\text{-}X}$-formula $\varphi = F(\Pi_2)\&F(\Pi_3)\&G(!\Pi_4)\&(!\Pi_2 \cup \Pi_1)\&F(\Pi_5\&\Pi_6)$. Basically, the formula imposes that:

- Regions Π_2 and Π_3 have to be eventually visited;
- Region Π_4 should be avoided at all times;
- Region Π_2 can be visited only after Π_1 was visited;
- The (disjoint) regions Π_5 and Π_6 should be eventually occupied at the same time (by both robots).

All the simulations carried out for this study have been performed on an Intel Intel(R) Core(TM) i5-6600 CPU at 3.30GHz with 16GB of RAM memory.

The solution was computed by performing the following steps: the $LTL_{\text{-}X}$-formula φ is translated into a Büchi automaton B with 8 states; the transition system of each robot (T) is computed and has 50 states; the transition system T_T, modeling the team of robots, is computed in 5.35 seconds with $50^2 = 2500$ states; the product automaton $A = T_T \times B$ is created in 39.6 seconds having 20000 states; finally, by a minimum path algorithm on A, the paths of the robots are computed in 68.61 seconds, obtaining the robot trajectories illustrated in Fig. 3. This is one of the existing optimal solutions that solves the problem with the same number of cells crossed by the team to reach the objective. The dots along robot trajectories indicate positions where robots synchronize (by pausing their motion) when crossing to next trajectory cell.

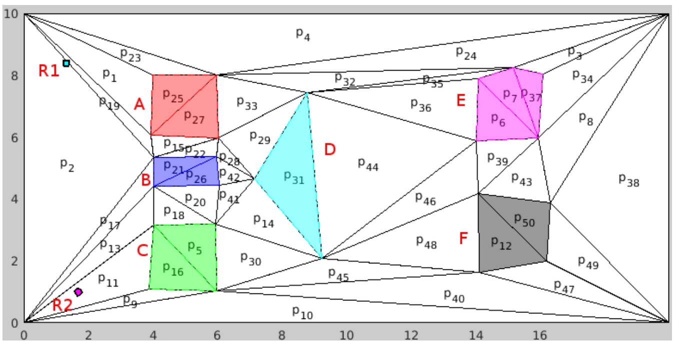


Fig. 2. Cell decomposition of the environment with 5 regions of interest, where $\Pi_{1..6}$ are labeled by characters A..F respectively. The team has two robots, initially placed in partition cells p_2 and p_{11}.

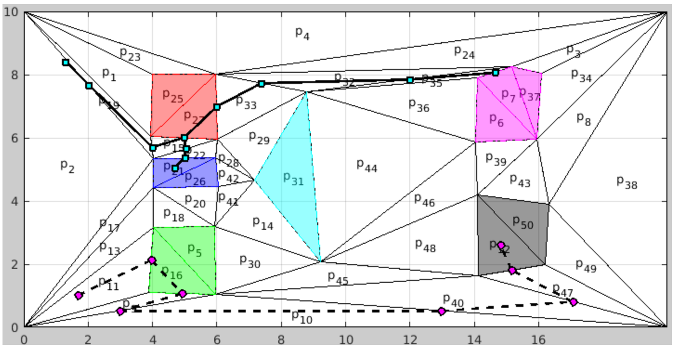


Fig. 3. Paths for the team of robots (R1,R2) fulfilling the $LTL_{\text{-}X}$-formula $\varphi = F(\Pi_2)\&F(\Pi_3)\&G(!\Pi_4)\&(!\Pi_2 \cup \Pi_1)\&F(\Pi_5\&\Pi_6)$.

3. BOOLEAN MISSIONS AND PETRI NET MODELS

The second method included in RMTool for multi-robot is the one described in (Mahulea and Kloetzer, 2014, 2016) and briefly presented in the following. In this case, the team mission is given as a Boolean formula instead of an $LTL_{\text{-}X}$ formula. It means that the temporal operators are not used, but only boolean ones (RMTool symbols): negation (!), disjunctions (|) and conjunction (&). Consequently, it is not possible to impose a specific sequencing or other temporal requirements for visiting regions.

The formula φ is defined over the set of variables $P_t \cup P_f$, where $P_t = \Pi = \{\Pi_1, \Pi_2, ..., \Pi_{|\Pi|}\}$ and $P_f = \{\pi_1, \pi_2, ..., \pi_{|\Pi|}\}$. P_t and P_f refers to the same regions of interest, but P_t suggests regions that should be visited along the trajectory, while P_f refers to the regions that should be visited in the last state of a run. Thus, the Boolean-based requirement allows the user to specify regions that can be traversed along robot trajectories, final regions that the robots should reach and never leave, and regions that have to be avoided along trajectories or in the final states.

The environment is partitioned as in the previous case by using a cell decomposition approach. Different than the case from Sec. 2 (where infinitely repeating sequences can be expressed), the robot trajectories from this section will have finite length in terms of visited cells. Moreover, the movement of the whole robotic team is modeled by means of a *Robot Motion Petri Net* (RMPN), whose topology does not vary with the team size. A RMPN is a 4-tuple $\mathcal{Q} = \langle \mathcal{N}, \mathbf{m}_0, \Pi, h \rangle$, where:

- $\mathcal{N} = \langle P, T, F \rangle$ is a Petri net where P is the set of places that represents the environment partition cells; T is the set of transitions representing the possibility of a robot to move between cells, i.e., the firing of a transition t corresponds to the movement of a robot from ${}^\bullet t = \{p_i\}$ to $t^\bullet = \{p_j\}$; $F \subseteq (P \times T) \cup (T \times P)$ is the set of unitary arcs that connect places and transitions;
- $\mathbf{m}_0$ is the initial marking, where each token represents a robot in a cell, i.e., $\mathbf{m}_0[p]$ is the number of robots located in the cell p at the initial state;
- $\Pi \cup \{\emptyset\}$ is the output alphabet, where $\{\emptyset\}$ denotes the empty observation;
- $h : P \to 2^{\Pi}$ is an observation map, such that if p_i has at least one token then observations from $h(p_i)$ are active.

The solution for the problem of finding the optimal paths for the team of robots is based on solving Integer Linear Programming (ILP) problems, with formulations detailed in (Mahulea and Kloetzer, 2014, 2016). The idea is to regard the team trajectory as crossing through a user-defined number (k) of intermediate configurations $\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_k$ (markings of $\mathcal{Q}$), where from m_{i-1} and m_i each robot moves at most one cell, $i = 1, \ldots, k$. Basically, k represents the maximum number of intermediate discrete states of each robot, besides its initial position from marking $\mathbf{m}_0$. This ensures that no spurious solutions (impossible firing sequences or cycles) are obtained for the RMPN. Theoretically, the upper-bound of k is $|T|$, but in practice much lower values of k suffice. Whenever the problem returns a solution, it is optimal, but if the value of k is too small, the problem becomes unfeasible. The intermediate configurations and the firing count vectors that link them are to be found by solving an ILP. The ILP constraints include the state equation of $\mathcal{Q}$, the formula φ and links between observations of $\mathcal{Q}$ and binary variables corresponding to atomic propositions from φ. The ILP objective is to minimize the total number of fired transitions (traversed cells) and to reduce the number of possible congestions (more robots in the same cell).

ILP problems belongs to the NP-hard complexity class and the computational cost is usually described by the number of unknowns and constraints. The solution developed for this problem has in this case a total number of $(k \times (3 \times (|P| + |T|) + 4 \times |\Pi| + |P| + 1)$ constrains and $(k \times (|P| + |T|) + 2 \times |\Pi| + 1)$ integer unknowns (corresponding to intermediate markings, firing count vectors, binary variables related to φ). As can be seen, the number of robots does not affect on the complexity of the ILP problem. This means that in the case of including more robots in the environment, the PN model does not change its topology. Therefore, in comparison with other approaches, computational advantages results especially for more robots, even though the RMPN model is used at this moment for less expressive specifications than in Sec. 2. For specifications and model assumed in this section, there are no robot synchronizations necessary during the team movement.

k	var	eq	ineq	ILP$_c$	ILP$_s$	R$_1$	R$_2$	R$_T$
10	1973	500	580	0.12	0.31	7	11	18
15	2953	750	830	0.27	5.38	12	6	18
20	3933	1000	1080	0.53	124.13	12	6	18

Table 1. Computation values for the full system approach. Number of variables is *var*, of equality constraints is *eq*, of inequality constraints is *ineq*. Time (in seconds) to construct the ILP problem is *ILP$_c$*, time to solve the ILP problem is *ILP$_s$*. *R1*, *R2*, *R$_T$* are the number of cells visited by the robot R$_1$, R$_2$ and by the team, respectively.

RMTool offers two possibilities to solve the planning problem for Boolean-based tasks: (i) the optimal solution that considers the full model $\mathcal{Q}$, as it has been described above; and (ii) a suboptimal solution based on a reduced team model. The reduction of the RMPN consists in lowering the number of places $|P|$ and transitions $|T|$. The idea is to iteratively combine places that correspond to adjacent cells satisfying the same region(s) of interest. By this second solution, the ILP formulation has fewer variables and constraints, but the solution is sub-optimal in terms of the number of traversed cells. The sub-optimal method reduces the upper-bound of k significantly, but a *lift* process is required for each transition fired in the reduced model, to find the sequence of transitions in the original $\mathcal{Q}$. This process involves solving a number of k Linear Programming Problems (LPPs).

Example 2. We use this method in the environment described in Ex. 1. The imposed formula is:

$$\varphi = \Pi_1 \& \pi_2 \& \Pi_3 \& !\Pi_4 \& \Pi_5 \& \Pi_6,$$

requiring that all regions except Π_4 are visited, and a robot ends its movement in region Π_2.The problem has been solved for different values of k, by the analysis of the full system $\mathcal{Q}$, as well as by using the reduced RMPN. The computational cost to create the PN system is under 1 second, having in the case of the full system $|P| = 50$ and $|T| = 146$, while the reduced system has $|P| = 7$ (one place for each region of interest, and one for the free space) and $|T| = 12$.

The resulting paths obtained by solving the problem on the full model $\mathcal{Q}$ for different values of k can be seen in Fig. 4 and Fig. 5. Table 1 includes the computation times for three values of k considered when solving this example. As observed in Table 1, the time required to find the solution increases considerably with the value of k, and, although the path for each robot changes, the total number of movements for the team of robots is the same, i.e., $k = 10$ yields the optimal solution. For $k < 10$ the ILP returns no solution, because the formula cannot be satisfied by less than 11 intermediate robot positions (k + initial position).

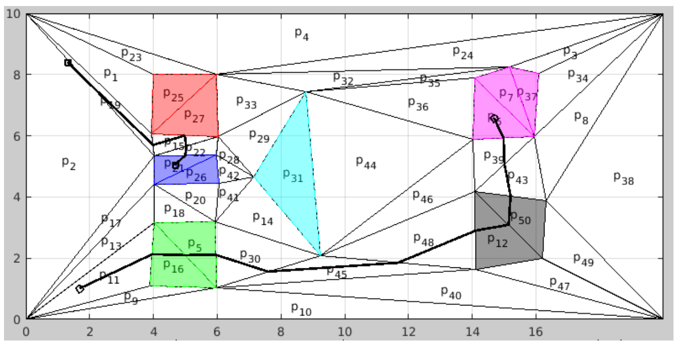


Fig. 4. Solution obtained when using the full model $\mathcal{Q}$ for $k = 10$

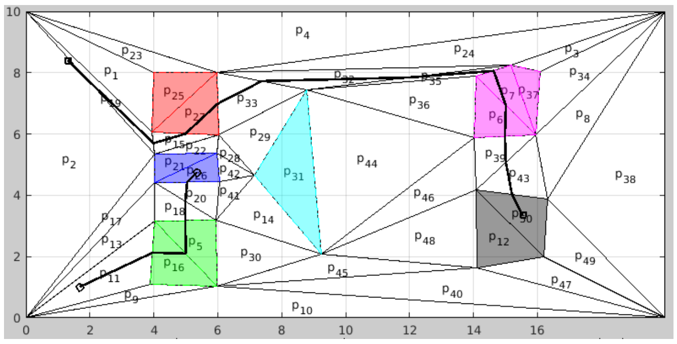


Fig. 5. Solution obtained on the full model $\mathcal{Q}$ for $k \geq 15$

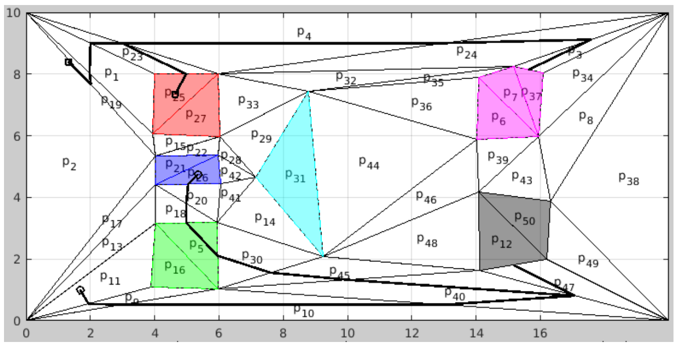


Fig. 6. Solution using the reduced system for $k = 5$

Solutions of the same problem when using the reduced system are shown in Fig. 6, 7 and 8, for different values of k. Table 2 collects relevant values of these simulations. Although the solutions are suboptimal, it can be noted that the computational overhead induced by solving the k LPPs is negligible, while the time for solving the ILP on the reduced model is significantly lower than the one for solving the ILP on the full model $\mathcal{Q}$. The obtained cost (number of robot movements) has been increased with 38% in the best case, i.e., $k = 15$ or $k = 20$, and with 55% in the worst case, i.e., $k = 10$.

4. EXAMPLES / COMPARISON

The planning methods briefly described in Sec. 2 and Sec. 3, and with detailed formal descriptions in works referenced therein, were integrated in our Matlab toolbox RMTool (`http://webdiis.unizar.es/RMTool/`). For this, various functionalities were included as separate functions for the involved operations (as construction of transition system or Petri net model for a team of robots,

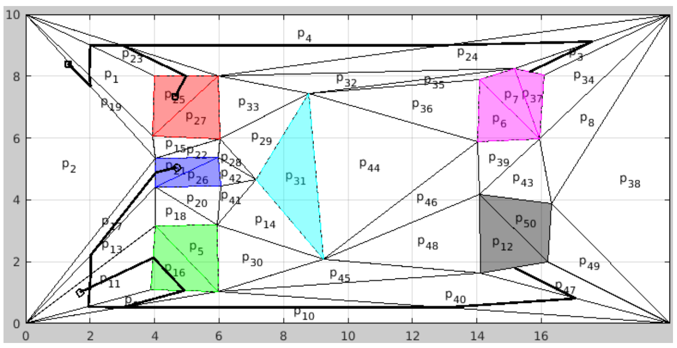


Fig. 7. Solution using the reduced system for $k = 10$

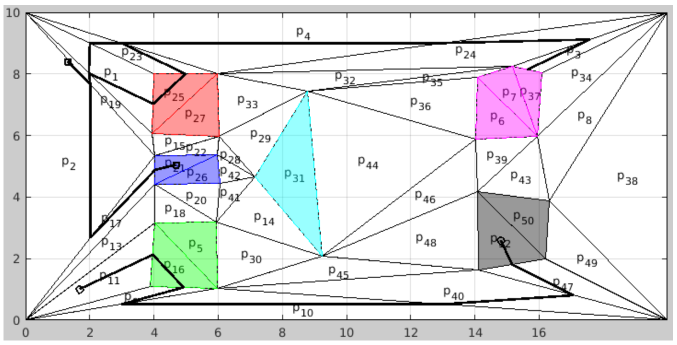


Fig. 8. Solution using the reduced system for $k \geq 15$

k	var	eq	ineq	ILP$_t$	LPP$_n$	LPP$_t$	R$_1$	R$_2$	R$_T$
5	108	35	72	0.05	5	0.12	13	14	27
10	203	70	107	0.03	10	0.13	13	15	28
15	298	105	142	0.03	15	0.14	18	7	25
20	393	140	177	0.03	20	0.13	18	7	25

Table 2. Relevant values for the solution obtained when using the reduced RMPN. $k = 5$ is the minimum value accepted to solve the problem. Number of variables is *var*, of equality constraints is *eq*, of inequality constraints is *ineq*. ILP$_t$ is the time required to solve the ILP problem, LPP$_n$ is the number of LPPs to solve, LPP$_t$ is the time required to solve all LPPs. *R1*, *R2*, R_T are the number of cells visited by the robot R$_1$, R$_2$ and by the team, respectively.

product automaton, ILP formulation etc.). The graphical interface of RMTool also provides quick access to these planning solutions. Additional software tools needed for running these functionalities include CDD - (Fukuda, 2016; Torrisi and Baotic, 2005) - for performing polyhedral operations useful in cell decompositions, LTL2BA - (Gastin and Oddoux, 2001) - for converting LTL formulas to Büchi automata, and Cplex - (IBM, 2016) - for solving ILPs and LPPs.

This section provides simulation results for the above presented path planning algorithms and outlines some advantages and limitations of each method.

Table 3 includes the size of involved transition systems and the computation times for solving an LTL mission in various scenarios. As expected, these numerical results concord with the size of A given in Sec. 2, in the sense of observing the drastic influence of the team size on the computation demands (rows 1-4 and 5-8). Note that, despite of size of A, the scenario from row 9 (i.e., 5 regions of interest, 2 robots and 9 states in B) is faster that the one from row 4 (i.e., environment with one region of interest, 4 robots and 2 states in B). This is mainly because most of the time is devoted to the construction of A, and in our implementation this construction is generally slowed down more by the size of T_T rather than the size of B.

Tables 4 and 5 show the results obtained when using Boolean missions and full, respectively reduced Petri net models. For allowing computation comparisons with LTL missions, the first 8 rows of Tables 4 and 5 refer to the same task as the first 8 rows of Table 3 (reachability of a region of interest). The scenario from the ninth row of Tables 4 and 5 is not as rich as the last one from Table 3, because of the restricted expressivity of Boolean missions when compared with LTL ones.

Computation times from Tables 3, 4, 5 suggest that, as long as the team specification can be expressed as a Boolean mission, one should use the approach from Sec. 3. However, if the mission has to include temporal specifications (e.g., order of visiting regions, synchronization when entering disjoint regions), one has to use the method from Sec. 2.

As expected from Sec. 3, Tables 4 and 5 show that the team size has negligible influence on the computation time when using Petri net team models. Although all times are small, one can observe that in some cases the reduced system (Table 5) is slower than the full one (Table 4). This is because of additional LPP problems that are solved for lifting the solution from the reduced model to the path in the full model. Thus, whenever the size of the full model $\mathcal{Q}$ is relatively small, there is no need to use the sub-optimal method based on the reduced system. However, for many cells in environment partition, the benefits of using the reduced system are two-folded: (1) the size of the model is reduced and (2) smaller values of parameter k can suffice for obtaining a solution. Both benefits (1) and (2) basically reduce the complexity of ILP problem, at the cost of obtaining a sub-optimal solution.

Unlike the method from Sec. 2, the computational time required by the approach based on Boolean missions and Petri net models can decrease when increasing the team size n. The intuition behind this is that for more robots there may be sufficient lower values of k (maximum number of cells traversed by a robot) in obtaining a solution. The influence of k on the size of the ILP problem was given in Sec. 3. The previous remark is exemplified in Table 6, when the same formula (visiting all regions of interest) is solved for one robot (rows 1,2) and for 10 robots (rows 3-7). As mentioned in Sec. 3, the number of robots does not afect to the size of the Mixed Integer Linear Programing (MILP) problem that solves the path planning for the team. Nonetheless in an environment divided in a big set of cells, the parameter k, i.e., the maximal number of cells that a robot need to cross to satisfy the formula, impacts the MILP problem. Because of that, in a mission where lots of regions of interest have to be visited, if the number of robots in the team is reduced, the min value for the parameter k should be bigger, hence the computational cost too. On the contrary if we introduce more robots to the team the min value of k is lower, thus the problem is solved in shorter time, being usefull this method for big teams of robots in cases where the computational cost is relevant, e.g., real time problems.

$\lvert\Pi\rvert$	R	T	T_T	t_{T_T}	B	A	t_A	t_{run}	t
1	1	8	8	0.02	2	16	0	0	0.03
1	2	8	64	0.02	2	128	0.1	0.03	0.06
1	3	8	512	0.35	2	1024	0.52	0.55	1.42
1	4	8	4096	108.88	2	8192	616.8	25.59	751.27
2	1	14	14	0.02	2	28	0	0.02	0.04
2	2	14	196	0.10	2	392	0.05	0.087	0.23
2	3	14	2744	7.37	2	5488	23.21	7.33	37.9
2	4	14	38416	19675.1	2	76832	118843	1515	140033.1
5	2	32	1024	1.03	9	9216	8.71	15.23	24.97

Table 3. Results of solving an LTL mission, with different numbers for regions of interest and for robots. For the first eight rows, the LTL formula is $\varphi = F(\Pi_1)$, and for the last row $\varphi = F(\Pi_1 \& \Pi_2) \& (!(\Pi_1 \& \Pi_2) \cup (\Pi_5)) \& (!(\Pi_1 \& \Pi_2) \cup (\Pi_4)) \& (!(\Pi_1 \& \Pi_2) \cup (\Pi_3))$. $\lvert\Pi\rvert$ is the number of regions of interest; R is the number of robots; T represents the number of environment cells; T_T is the number of states of the transition system for the team of robots; t_{T_T} is the time to compute T_T; B is the number of states of the Büchi automaton for φ; A is the number of states for the automaton used to find the robot paths; t_A is the time to create A; t_{run} is the time to find the path that satisfies the formula φ; t is the total computation time.

$\lvert\Pi\rvert$	R	k	Q	var.	eq.	ineq.	t
1	1	10	8P/20T	283	80	93	0.02
1	2	10	8P/20T	283	80	93	0.01
1	3	10	8P/20T	283	80	93	0.03
1	4	10	8P/20T	283	80	93	0.03
2	1	10	14P/38T	525	140	163	0.05
2	2	10	14P/38T	525	140	163	0.05
2	3	10	14P/38T	525	140	163	0.08
2	4	10	14P/38T	525	140	163	0.06
5	2	11	32P/92T	1375	352	409	0.15

Table 4. Results for solving a Boolean mission with full Petri net models. For the first eight rows the task is $\varphi = \Pi_1$, and for the last row $\varphi = \pi_1 \& \pi_2 \& \Pi_3 \& \Pi_4 \& \Pi_5$. $\lvert\Pi\rvert$ is the number of regions of interest; R is the number of robots; k is the maximum number of intermediate markings; Q is the size of Petri net model, in form (number of places / number of transitions); var./eq./ineq. are the number of variables/equality constraints/inequality constraints of the ILP problem; t is the time required to solve the problem.

5. CONCLUSION

This work reports recent enhancements that were included in the RMTool, a MATLAB toolbox with an interactive graphical user interface dedicated to mobile robot path planning. These enhancement refer to planning the movement of a team of robots, starting from a requirement formulated over a set of regions of interest defined in the environment. The team mission formalism belongs to one

$\lvert\Pi\rvert$	R	k	Q	var.	eq.	ineq.	t
1	1	10	2P/2T	43	20	27	0.02
1	2	10	2P/2T	43	20	27	0.03
1	3	10	2P/2T	43	20	27	0.03
1	4	10	2P/2T	43	20	27	0.06
2	1	10	3P/4T	75	30	42	0.03
2	2	10	3P/4T	75	30	42	0.04
2	3	10	3P/4T	75	30	42	0.01
2	4	10	3P/4T	75	30	42	0.02
5	2	10	6P/10T	171	60	91	0.12

Table 5. Results for solving the same tasks as in Table 4, by using reduced Petri net models. The header row contains the same information as in Table 4.

$\lvert\Pi\rvert$	R	k	Q	var.	eq.	ineq.	R_T	Fval.	t
10	1	32	66P/194T	8341	2112	2228	32	33	140.8
10	1	33	66P/194T	8601	2178	2294	32	33	120.1
10	10	10	66P/194T	2621	660	776	36	35	0.42
10	10	11	66P/194T	2881	726	842	36	35	1.22
10	10	12	66P/194T	3141	792	908	38	33	1.86
10	10	13	66P/194T	3401	858	974	37	31	3.55
10	10	14	66P/194T	3661	924	1040	36	28	2.38

Table 6. Results for solving the same Boolean mission for different number of robots. The task is $\varphi = \Pi_1 \& \Pi_2 \& \Pi_3 \& \Pi_4 \& \Pi_5 \& \Pi_6 \& \Pi_7 \& \Pi_8 \& \Pi_9 \& \Pi_{10}$. The header row contains the same data as in Table 4, except R_T - the number of movements performed by the team of robots, and Fval. - objective value to optimize, that considers robot movements and congestions during the trajectories.

of two classes: LTL specifications or Boolean-based ones. For LTL tasks, previous researches propose algorithmic solutions based on transition system models, while for the less expressive Boolean tasks there are computationally attractive results based on Petri net models. The current paper describes these existing approaches and reports their integration into RMTool, with the purpose of providing access to these methods without necessitating Matlab programming knowledge or deep understanding of the involved formal tools.

REFERENCES

Baier, C. and Katoen, J.P. (2008). *Principles of Model Checking*. MIT Press, Boston.

Belta, C. and Habets, L.C. (2006). Controlling a class of nonlinear systems on rectangles. *IEEE Transactions on Automatic Control*, 51(11), 1749–1759.

Choset, H.M. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press.

Corke, P. (2011). *Robotics, Vision and Control. Fundamental algorithms in Matlab*. Springer.

Corke, P.I. (1996). A robotics toolbox for Matlab. *IEEE Robotics & Automation Magazine*, 3(1), 24–32.

Couceiro, M.S. (2012). MRSim- multi-robot simulator. URL `http://home.isr.uc.pt/~micaelcouceiro/media/help/helpMRSim.htm`.

De Berg, M., Van Kreveld, M., Overmars, M., and Schwarzkopf, O.C. (2000). Computational geometry. In *Computational geometry*, 1–17.

Ding, X., Kloetzer, M., Chen, Y., and Belta, C. (2011). Automatic deployment of robotic teams. *IEEE Robotics and Automation Magazine*, 18(3), 75–86.

Dudek, G. and Jenkin, M. (2010). *Computational principles of mobile robotics*. Cambridge university press.

Fainekos, G.E., Girard, A., Kress-Gazit, H., and Pappas, G.J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, 45(2), 343–352.

Fukuda, K. (2016). CDD/CDD+ package. *http://www.inf.ethz.ch/personal/fukudak/cdd_home/*.

Gastin, P. and Oddoux, D. (2001). Fast LTL to Büchi automata translation. In *International Conference on Computer Aided Verification*, 53–65.

Gonzalez, R., Mahulea, C., and Kloetzer, M. (2015). A Matlab-based interactive simulator for mobile robotics. In *CASE 2015 IEEE International Conference on Automation Science and Engineering*, 310–315.

Guo, M., Tumova, J., and Dimarogonas, D. (2014). Cooperative decentralized multi-agent control under local LTL tasks and connectivity constraints. In *IEEE Conference on Decision and Control*, 75–80.

Hrabeck, J. (2001). Autonomous mobile robotics toolbox, SIMROBOT. URL `http://serdis.dis.ulpgc.es/~ii-srm/MatDocen/notas_practicas/Simrobot/simrobot_en.html`.

IBM (2016). IBM ILOG CPLEX optimization studio - software. URL `http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/`.

Kloetzer, M. and Mahulea, C. (2014). A Petri net based approach for multi-robot path planning. *Discrete Event Dynamic Systems: Theory and Applications*, 24(4), 417–445.

Kloetzer, M. and Mahulea, C. (2012). LTL planning in dynamic environments. In *WODES 2012: 11th International Workshop on Discrete Event Systems*, 294–300.

Kloetzer, M. and Mahulea, C. (2015). LTL-based planning in environments with probabilistic observations. *IEEE Transactions on Automation Science and Engineering*, 12(4), 1407–1420.

Kloetzer, M. and Mahulea, C. (2016). Multi-robot path planning for syntactically co-safe LTL specifications. In *WODES 2016: 13th International Workshop on Discrete Event Systems*, 452–458.

LaValle, S.M. (2006). *Planning algorithms*. Cambridge university press.

Mahulea, C. and Kloetzer, M. (2014). Planning mobile robots with boolean-based specifications. In *CDC'2014: 53rd IEEE Conference on Decision and Control*. Los Angeles, USA.

Mahulea, C. and Kloetzer, M. (2016). Robot planning based on boolean specifications using Petri net models. Under review.

Siegwart, R., Nourbakhsh, I.R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

Torrisi, F. and Baotic, M. (2005). Matlab interface for the CDD solver. *http://control.ee.ethz.ch/~hybrid/cdd.php*.

Wolper, P., Vardi, M.Y., and Sistla, A.P. (1983). Reasoning about infinite computation paths. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, 185–194.