Multi-Objective Search for Optimal Multi-Robot Planning with Finite LTL Specifications and Resource Constraints

Philipp Schillinger[1,2], Mathias Bürger[1] and Dimos V. Dimarogonas[2,3]

***Abstract*— We present an efficient approach to plan action sequences for a team of robots from a single finite LTL mission specification. The resulting execution strategy is proven to solve the given mission with minimal team costs, e.g., with shortest execution time. For planning, an established graph-based search method based on the multi-objective shortest path problem is adapted to multi-robot planning and extended to support resource constraints. We further improve planning efficiency significantly for missions which consist of independent parts by using previous results regarding LTL decomposition. The efficiency and practicality of the ROS implementation of our approach is demonstrated in example scenarios.**

I. Introduction

Linear Temporal Logic (LTL) has been established as a mean to formally specify requirements and missions for robots and autonomous systems [1, 2]. Being originally developed for model checking [3], LTL integrates well with planning of execution strategies. In a multi-robot setting, [4] proposes a bottom-up approach to plan actions, given an LTL specification for each robot. [5] extends the vehicle routing problem with LTL constraints and plans a solution based on Mixed-Integer Linear Programming (MILP). [2, 6] assume a single mission for the robotic team and employ trace-closed languages to distribute the mission. A main challenge in planning for multi-robot systems remains the computational complexity. To reduce complexity, [7] abstracts from independent motions of the single agents. In previous work [8], we proposed a way to identify independent parts of a finite LTL mission. The named approaches focus on LTL constraints, which are discrete in nature. However, more complex missions usually need to deal with additional constraints on resources. Reactive approaches [1, 9] can be used to handle resource constraints online. For cost-optimal strategies, it is required to consider these constraints already during planning.

This paper proposes a novel multi-robot LTL planning approach with costs and resource constraints by formulating it as a multi-objective search problem [10]. Multi-objective planning is well established in the area of operations research and includes search methods based on label setting [11, 12] and label correcting [13]. Although assignment approaches [14, 15] exist to plan allocation if execution costs of the tasks are known, this is not the case here and execution needs to planned as well. An established method, which is proven to find all Pareto optimal solutions of a multi-criteria search problem, is Martins' algorithm [16]. In this context, resource constraints can be modeled efficiently as additional objectives of the search [17, 18].

Fig. 1. Two robots operating in an office environment based on the ROS implementation of our proposed planning approach.

The contributions of this paper are as follows: (1) We formulate a multi-agent team cost-optimal planning problem with resource constraints based on system models and a finite LTL mission specification for the team in Sections II and III. (2) By extending Martins' algorithm, we propose an efficient algorithm to solve this planning problem in Section IV. (3) As a special case of representing a team, we build upon our previous results of identifying independent parts of a mission [8] and outline an adaptation in Section V to improve efficiency for decomposable missions. (4) We present a case study of our ROS implementation in Section VI and discuss it for variations of an example mission.

II. Preliminaries

LTL formulas φ extend Boolean operators like $\neg$ "not", $\wedge$ "and", and $\vee$ "or" by temporal operators like $\circ$ "next", $\Diamond$ "eventually", $\Box$ "always", and $\mathcal{U}$ "until". Consequently, LTL formulas are not evaluated over a single set of *atomic propositions* $\pi \in \Pi$ where π can either be $\top$ "true" or $\bot$ "false", but instead over a sequence $\sigma \colon \mathbb{N} \to 2^{\Pi}$ of such sets where $\sigma(t) \subseteq \Pi$ contains all propositions which are true at time t. A sequence σ is said to *satisfy* an LTL formula φ, denoted by $\sigma \vDash \varphi$, according to certain semantics as for example presented in [3]. Specifically, for any LTL formulas φ_1 and φ_2, the following semantics hold.

[1]Bosch Center for Artificial Intelligence, Renningen, Germany. {philipp.schillinger, mathias.buerger}@de.bosch.com

[2]KTH Centre for Autonomous Systems and ACCESS Linnaeus Center, EES, KTH Royal Institute of Technology, Stockholm, Sweden. {schillin, dimos}@kth.se

[3]Supported by the H2020 ERC Starting Grant BUCOPHSYS, the Swedish Research Council (VR), the Swedish Foundation for Strategic Research (SSF), and the Knut och Alice Wallenberg Foundation (KAW).

This work was supported by the EU H2020 Research and Innovation Programme under GA No. 731869 (Co4Robots).

– $\sigma \vDash \circ\varphi_1$ iff $\sigma_{[1,\dots]} \vDash \varphi_1$
– $\sigma \vDash \Diamond\varphi_1$ iff there exists a t_1 such that $\sigma_{[t_1,\dots]} \vDash \varphi_1$
– $\sigma \vDash \Box\varphi_1$ iff for all t_1 it holds that $\sigma_{[t_1,\dots]} \vDash \varphi_1$
– $\sigma \vDash \varphi_1 \,\mathcal{U}\, \varphi_2$ iff there exists a t_2 such that $\sigma_{[t_2,\dots]} \vDash \varphi_2$ and $\sigma_{[t_1,\dots]} \vDash \varphi_1$ for all $t_1 < t_2$

with $\sigma_{[t,\dots]}$ denoting the subsequence starting at t. These sequences σ can be infinite, although classes of so-called finite LTL specifications [19], e.g., *co-safe LTL* [20], can be satisfied by finite sequences. A finite LTL formula ϕ can be translated into a finite automaton [3] such that the automaton accepts a sequence σ if and only if it satisfies the corresponding LTL formula.

Definition 1 (NFA)**.** *A nondeterministic finite automaton (NFA) is given as the tuple $\mathcal{F} := (Q, Q_0, \alpha, \delta, F)$ consisting of **(1)** a set of states Q, **(2)** a set of initial states $Q_0 \subseteq Q$, **(3)** an alphabet α of Boolean formulas over $\pi \in \Pi$, **(4)** a set of transition conditions $\delta\colon Q \times Q \to \alpha$, **(5)** a set of accepting (final) states $F \subseteq Q$.*

Each sequence σ describes a sequence of states $q \in Q$, called a *run* $\rho\colon \mathbb{N} \to Q$, such that ρ starts in an initial state $q_0 \in Q_0$ and the Boolean transition conditions are satisfied $\sigma(t) \vDash \delta(\rho(t-1), \rho(t))$ for all t. A run ρ is called *accepting*, meaning that σ is accepted by the automaton, in the case that ρ ends in an accepting state $q_n \in F$.

III. System Model

In this paper, we study a cost-optimal planning problem for a team of mobile agents with limited resources and an LTL mission specification. An agent is a mobile system in a given environment with additional actuation capabilities. Formally, it is represented by a transition system that combines a topological map of the environment with discrete actions, which can be executed at certain locations.

Definition 2 (Agent Model)**.** *An agent model is given as the transition system $\mathcal{A} := (S_\mathcal{A}, s_{0,\mathcal{A}}, A_\mathcal{A}, \Pi, \lambda)$ consisting of **(1)** a set of states $S_\mathcal{A}$, **(2)** an initial state $s_{0,\mathcal{A}} \in S_\mathcal{A}$, **(3)** a set of actions $A_\mathcal{A} \subseteq S_\mathcal{A} \times S_\mathcal{A}$, **(4)** a set of propositions Π, **(5)** a labeling function $\lambda\colon S_\mathcal{A} \to 2^\Pi$.*

For a finite LTL mission specification, a product of the agent model $\mathcal{A}$ and the NFA $\mathcal{F}$ can be formulated.

Definition 3 (Product Automaton)**.** *A product automaton is a tuple $\mathcal{P} = \mathcal{F} \otimes \mathcal{A} := (S_\mathcal{P}, S_{0,\mathcal{P}}, A_\mathcal{P})$ consisting of **(1)** a set of states $S_\mathcal{P} = Q \times S_\mathcal{A}$, **(2)** a set of initial states $S_{0,\mathcal{P}} = \{(q, s_{0,\mathcal{A}}) \in S_\mathcal{P}\colon q \in Q_0\}$, **(3)** a set of actions $A_\mathcal{P} = \{((q_\mathrm{s}, s_\mathrm{s}), (q_\mathrm{t}, s_\mathrm{t})) \in S_\mathcal{P} \times S_\mathcal{P}\colon (s_\mathrm{s}, s_\mathrm{t}) \in A_\mathcal{A} \wedge \lambda(s_\mathrm{s}) \vDash \delta(q_\mathrm{s}, q_\mathrm{t})\}$.*

A run ending in a state $s_n = (q, s_\mathcal{A})$ with $q \in F$ being an accepting state in $\mathcal{F}$ determines an action sequence

$$\beta = s_0 a_1 s_1 \dots a_n s_n \tag{1}$$

with $s_i \in S_\mathcal{P}$, $s_0 \in S_{0,\mathcal{P}}$, and $a_j = (s_{j-1}, s_j) \in A_\mathcal{P}$. If β is executed by the agent, the LTL specification is fulfilled.

As this paper addresses a multi-agent planning problem with N agents, it is required to construct a joint *team automaton* $\mathcal{C} := (S_\mathcal{C}, S_{0,\mathcal{C}}, F_\mathcal{C}, A_\mathcal{C})$ where $S_\mathcal{C}$ is a set of states, $S_{0,\mathcal{C}} \subseteq S_\mathcal{C}$ a set of initial states, $F_\mathcal{C} \subseteq S_\mathcal{C}$ a set of final states, and $A_\mathcal{C} \subseteq S_\mathcal{C} \times S_\mathcal{C}$ a set of actions. An action sequence β defined over $\mathcal{C}$ instead of $\mathcal{P}$ describes a sequence for the team.

There are several possibilities to construct a team automaton. The most straightforward would be a product automaton of all N agents' automata. In this case, a state in the team automaton is a tuple of all individual states of all agents. Since this means that the state space is exponential in N, this approach is only feasible for small team sizes. An alternative team model representation was proposed in [8] which is linear in N. We recall this representation in Section V and discuss implications for the presented planning approach, although the general planning algorithm can be used for any construction of a team model $\mathcal{C}$.

In order to decide which action sequence β should be preferred, we associate non-negative costs $C\colon A_\mathcal{C} \to \mathbb{R}_{\geq 0}$ with each action $a \in A_\mathcal{C}$ of $\mathcal{C}$. Please note that in the context of this paper, we assume that an action is always performed by a single agent and a cost only occurs for this agent. Each action sequence β for the whole team is associated with an N-dimensional cost vector $c_\beta \in \mathbb{R}^N_{\geq 0}$ where each component $c_{\beta,r}$ represents the costs associated with the respective agent $r \in \{1, \dots, N\}$. Formally, c_β is given by

$$c_\beta = \sum_{a \in \beta} C(a) \mathrm{e}_\mathrm{a} \tag{2}$$

where $\mathrm{e}_\mathrm{a} \in \{0,1\}^N$ denotes the N-dimensional unit vector which identifies the agent r associated with the action a.

As c_β specifies a multi-dimensional cost, an overall team cost $\kappa\colon \mathbb{R}^N_{\geq 0} \to \mathbb{R}_{\geq 0}$ needs to be defined. The following parametrized team cost is shown to be especially relevant in the context of mobile multi-agent systems:

$$\kappa(c_\beta) = (1 - \epsilon) \cdot \|c_\beta\|_\infty + \epsilon \cdot \|c_\beta\|_1 \tag{3}$$

with $\epsilon \in (0, 1]$. The parameter ϵ determines a trade-off between the maximal agent costs $\|c_\beta\|_\infty$ and the sum of all actions regardless of which agent performs it $\|c_\beta\|_1$. As action costs $C(a)$ often correspond to approximate execution times, we choose ϵ close to zero with the goal to minimize the execution time of the mission. For technical reasons discussed later in the proof of Lemma 1, ϵ still needs to be strictly greater than zero. As such, $\|c_\beta\|_1$ can be interpreted as a regularization term to penalize unnecessary actions of the agents.

The resource constraints are modeled as inequality constraints on a set of M different resource variables. Analogously to costs, the resource consumption of an action $a \in A_\mathcal{C}$ is specified by the vector $\Gamma\colon A_\mathcal{C} \to \mathbb{R}^M$. We require that the resources available after an action sequence β are strictly positive, that is

$$\gamma_\beta = \gamma_0 + \sum_{a \in \beta} \Gamma(a) > 0 \tag{4}$$

where γ_β is the status of the resources after the action sequence and $\gamma_0 \in \mathbb{R}^M_{\geq 0}$ denotes the initial resources. The

inequality $\gamma_\beta > 0$ is defined as an inequality $\gamma_{\beta,i} > 0$ on each component $\gamma_{\beta,i}$ with $i \in \{1, \dots, M\}$. Note that constraints with $\gamma_{\beta,i} \geq 0$ can be represented as well. Although $\gamma_{\beta,i}$ is continuous, we assume a smallest possible change $\gamma_{\Delta,i}$ between any two actions $a_j, a_k \in A_\mathcal{C}$, given by $\gamma_{\Delta,i} = \min_{(a_j,a_k)} |\Gamma(a_j)_i - \Gamma(a_k)_i|$. Consequently, $\gamma_{\beta,i} \geq 0$ can be written as $\gamma_{\beta,i} + \gamma_{\Delta,i} > 0$.

Finally, the planning problem can be summarized as follows. Given a team model $\mathcal{C}$ constructed from individual product automata $\mathcal{P}$ for an LTL mission specification $\mathcal{M}$, find an action sequence β_{opt} with minimal team cost $\kappa(c_\beta)$ such that $\mathcal{M}$ is fulfilled and all constraints $\gamma_\beta > 0$ are satisfied.

IV. Planning

The planning problem specified above cannot be handled as a classical shortest path problem due to the structure of κ resulting from the multi-agent setting, in particular minimizing the maximal team cost. Consequently, search algorithms like Dijkstra or Bellman-Ford will fail to find the action sequence β which minimizes $\kappa(c_\beta)$. Instead, we propose to consider the multi-agent planning problem as a multi-objective planning problem with the objectives c_β and the goal to minimize the team cost $\kappa(c_\beta)$. In particular, using multi-object planning opens also the possibility to directly handle resource constraints as discussed later in Lemma 3.

Algorithm 1 outlines the proposed planning approach. It is an adaptation of the Martins' algorithm [16], an established *label-setting* approach used for multi-objective planning in operations research. We transfer this algorithm to constrained multi-agent mission planning by choosing a suitable cost function, allowing early termination, and adding resource constraints, as discussed in the following.

The idea of the classical Martins' algorithm is similar to the single-objective shortest path search proposed by Dijkstra. But instead of operating directly on the states of the graph, each state is assigned a set of labels of which the cost vectors are all Pareto optimal and the classical Martins' algorithm finds all Pareto optimal paths described by these labels. The algorithm distinguishes for each state $s \in S_\mathcal{C}$ between a set of temporary labels $L_{\text{t},s}$ which are candidate optimal labels during planning time, and permanent labels $L_{\text{p},s}$ which form the final set of Pareto optimal labels. Each label $l = (c_\beta^{(l)}, \gamma_\beta^{(l)}, v, i_v)$ of s not only consists of a cost vector $c_\beta^{(l)}$ and a resource vector $\gamma_\beta^{(l)}$ of a certain action sequence β, but also refers a predecessor state $v \in S_\mathcal{C}$ of s and a label index $i_v \in \{1, \dots, \text{card}(L_{\text{p},v})\}$ to one of v's permanent labels, called predecessor label, with $\text{card}(L_{\text{p},v})$ denoting the number of permanent labels at state v.

Given a label l at state s, it is possible to construct an action sequence β leading to s and resulting in the costs $c_\beta^{(l)}$ and resources $\gamma_\beta^{(l)}$ by tracing all predecessor labels backwards based on their respective predecessor state v and index i_v. Consequently, we will abbreviate $c_\beta^{(l)}$ with $c^{(l)}$ and $\gamma_\beta^{(l)}$ with $\gamma^{(l)}$ in the following whenever c_β, γ_β refer to a specific label from which β is constructed.

Algorithm 1 Constrained Optimal Multi-Agent Planning

Input: Team model $\mathcal{C}$, team cost function κ, resources γ_0
Output: Optimal final label l_{fin} to construct actions β_{fin}

Notation Remarks:
$S_\mathcal{C}$ ($S_{0,\mathcal{C}}, F_\mathcal{C}$), $A_\mathcal{C}$ – states (initial, final) and actions of $\mathcal{C}$
$l = (c^{(l)}, \gamma^{(l)}, v, i_v)$ – label with costs, resources, predecessor state, and label index at predecessor
$L_{\text{t},s}$ ($L_{\text{p},s}$) – set of temporary (permanent) labels $\forall s \in S_\mathcal{C}$
$l <_\text{P} \ell$ – short notation for $(c^{(l)}, -\gamma^{(l)}) <_\text{P} (c^{(\ell)}, -\gamma^{(\ell)})$

1: $L_{\text{t},v} \leftarrow \{([0, \dots, 0]^T, \gamma_0, \varnothing, \varnothing)\}, \forall v \in S_{0,\mathcal{C}}$
2: $L_{\text{t},s} \leftarrow \emptyset, \forall s \in S_\mathcal{C} \setminus S_{0,\mathcal{C}}$
3: $L_{\text{p},s} \leftarrow \emptyset, \forall s \in S_\mathcal{C}$
4: **while** $\forall s \in S_\mathcal{C} \colon L_{\text{t},s} \neq \emptyset$ **do**
▶ Find label l with lowest costs and make it permanent
5: $(s, l) \leftarrow \text{argmin}_{s \in S_\mathcal{C}, l \in L_{\text{t},s}} \{\kappa(c^{(l)})\}$
6: $L_{\text{t},s} \leftarrow L_{\text{t},s} \setminus \{l\}$
7: $L_{\text{p},s} \leftarrow L_{\text{p},s} \cup \{l\}$
▶ Terminate search if any final state is reached
8: **if** $s \in F_\mathcal{C}$ **then return** $l_{\text{fin}} \leftarrow l$ ▷ Best found first
▶ Calculate labels for all successors v of s
9: **for all** $v \in S_\mathcal{C} \colon a = (s, v) \in A_\mathcal{C}$ **do**
10: $c_{\text{new}} \leftarrow c^{(l)} + C(a) \cdot \text{e}_\text{a}$
11: $\gamma_{\text{new}} \leftarrow \gamma^{(l)} + \Gamma(a)$
12: $\ell \leftarrow (c_{\text{new}}, \gamma_{\text{new}}, s, i_s)$ with $i_s = \text{card}(L_{\text{p},s})$
▶ Only add and keep non-dominated temporary labels
13: **if** $\gamma_{\text{new}} > 0 \wedge \nexists l \in L_{\text{t},v} \cup L_{\text{p},v} \colon l <_\text{P} \ell$ **then**
14: $L_{\text{t},v} \leftarrow L_{\text{t},v} \setminus \{l \in L_{\text{t},v} \colon \ell <_\text{P} l\}$
15: $L_{\text{t},v} \leftarrow L_{\text{t},v} \cup \{\ell\}$
16: **return** $l_{\text{fin}} \leftarrow \varnothing$ ▷ No final state reachable

In each iteration (line 4), we choose one candidate label l from $L_{\text{t},s}$ of all states $s \in S_\mathcal{C}$ such that its cost vector $c^{(l)}$ minimizes the team cost function κ and extend the action sequence to l by possible next actions (line 9). The operator $<_\text{P}$ denotes a "less than"-relation in the Pareto sense, i.e., $(a_1, \dots, a_n)^T <_\text{P} (b_1, \dots, b_n)^T \Leftrightarrow a \neq b \wedge a_i \leq b_i, \forall i \in \{1, \dots, n\}$. In the case that $a <_\text{P} b$, we say that *a dominates* b which means b cannot lead to a better result than a.

Since Martins' algorithm belongs to the class of *label-setting* algorithms, it is crucial that temporary labels $l \in L_{\text{t},s}$ are selected in the correct order. Specifically, it is required that l is only made permanent (line 7) if it is guaranteed to be Pareto optimal. Given a certain cost function, in the following denoted by $h \colon \mathbb{R}^N \to \mathbb{R}$, [11] defines two properties of h to decide if h selects labels in the correct order. First, *dominance* defined as $c^{(l)} <_\text{P} c^{(\ell)} \implies h(c^{(l)}) < h(c^{(\ell)})$ and second, *monotonicity* defined as $h(c^{(l)}) \leq h(c^{(\ell)})$ for all ℓ referring to l as predecessor label. Theorem 4.1 in [11] then states that h selects labels in the correct order if it fulfills both properties.

To apply Martins' algorithm to the domain of constrained multi-agent planning, we make three adaptations. First, we choose the cost function $h := \kappa$ (line 5) to minimize our team cost function and show that κ is a valid choice.

Lemma 1 (Team Cost Optimization)**.** *κ is a cost function which selects labels in the correct order for $\epsilon \in (0, 1]$ and non-negative action costs.*

Proof. As shown by [11] for non-negative costs, $\|c_\beta\|_1$ fulfills both *dominance* and *monotonicity* while $\|c_\beta\|_\infty$ fulfills *monotonicity*. $\|c_\beta\|_\infty$ does not satisfy *dominance* since a dominated cost vector $c^{(l)} <_\mathrm{P} c^{(\ell)}$ could lead to the same $\|c^{(l)}\|_\infty = \|c^{(\ell)}\|_\infty$. However, $c^{(l)} <_\mathrm{P} c^{(\ell)}$ implies that $\|c^{(l)}\|_1 < \|c^{(\ell)}\|_1$ and consequently $\kappa(c^{(l)}) = (1-\epsilon) \cdot \|c^{(l)}\|_\infty + \epsilon \cdot \|c^{(l)}\|_1 < (1-\epsilon) \cdot \|c^{(\ell)}\|_\infty + \epsilon \cdot \|c^{(\ell)}\|_1 = \kappa(c^{(\ell)})$ for $\epsilon > 0$ and $(1-\epsilon) \geq 0$ fulfills both *dominance* and *monotonicity*. □

Second, we terminate the search (line 8) as soon as an accepting state is reached. This is possible because we are only interested in an accepting path β which minimizes $\kappa(c_\beta)$ instead of searching all Pareto optimal paths and as shown in the following, $\kappa(c_\beta)$ is minimal for the first β.

Lemma 2 (Early Termination)**.** *The first feasible solution l_fin found by Algorithm 1 has minimal team cost $\kappa(c_\beta^{(\mathrm{fin})})$.*

Proof. As shown in Lemma 1, the costs $c^{(l)}$ of a chosen label l cannot be dominated by the costs $c^{(\ell)}$ of another label ℓ since labels are guaranteed to be selected in the correct order and $\kappa(c^{(\ell)}) < \kappa(c^{(l)})$ only if $c^{(\ell)} <_\mathrm{P} c^{(l)}$. □

Third, we add constraints γ_β to the algorithm and update γ_β for each action similar to updating the costs c_β. Also, we extend the check for Pareto optimality $l <_\mathrm{P} \ell := (c^{(l)}, -\gamma^{(l)}) <_\mathrm{P} (c^{(\ell)}, -\gamma^{(\ell)})$ in lines 12 and 13 by $-\gamma_\beta$.

Lemma 3 (Resource Constraints)**.** *All action sequences β_fin to l_fin found by Algorithm 1 respect the resource constraints $\gamma_\beta > 0$. Additionally, any Pareto optimal action sequence β_fin which respects $\gamma_\beta > 0$ can be found.*

Proof. Labels ℓ are only added if no constraints are violated, given by $\gamma_\mathrm{new} > 0$ (line 13). Consequently, all sequences β can only contain actions that do not violate any constraints.

Furthermore, extension of the Pareto optimality check of labels by $-\gamma_\beta$ ensures that no label l is considered to be dominated by another label ℓ in the case that $c^{(\ell)} <_\mathrm{P} c^{(l)}$, but with $-\gamma^{(l)} <_\mathrm{P} -\gamma^{(\ell)}$. If $c^{(\ell)} = c^{(l)}$ while still $-\gamma^{(l)} <_\mathrm{P} -\gamma^{(\ell)}$, l dominates ℓ because $\gamma^{(\ell)} > 0$ only if $\gamma^{(l)} > 0$. □

Finally, we can show that Algorithm 1 finds the optimal action sequence for the team of agents, if one exists. For the termination of the algorithm, cycles in $\mathcal{C}$ require special attention. Considering that the resource modification $\Gamma(a)$ of an action a can be positive, cycles can provide a way to improve resources infinitely often and thus, create an infinite set of Pareto optimal labels.

Theorem 1 (Constrained Optimal Multi-Agent Planning)**.** *Assume non-negative costs and that no cycles improve resources with zero cost. If there exists a set of action sequences β such that all constraints $\gamma_\beta > 0$ are satisfied, Algorithm 1 finds one of these action sequences β_opt with minimal team cost $\kappa(c_{\beta_\mathrm{opt}})$.*

Proof. As shown by Lemma 3, Algorithm 1 can find any feasible action sequence and it will only find these. Note that, in the excluded case that cycles improve resources with zero cost, the algorithm would get stuck by improving resources infinitely often. Finally, as shown by Lemma 2, the first solution β_fin found by Algorithm 1 is the one with optimal team costs $\beta_\mathrm{opt} = \beta_\mathrm{fin}$. □

If no action sequence β exists such that $\gamma_\beta > 0$, Algorithm 1 will terminate with an empty result, given that no further non-dominated actions are found. Note that, if $\mathcal{C}$ contains cycles which improve resources infinitely often, the set of non-dominated actions is infinite and the algorithm would not terminate. This can be addressed by restricting resources γ_β to a certain range such that $\gamma_\beta < \gamma_\mathrm{max}$ for an upper bound γ_max, or by specifying a suitable abortion criterion like maximum cost or planning time.

V. Mission Decomposition

Our previous work [8] addresses the case that a mission implicitly consists of multiple independent parts, called *tasks* $\mathcal{T}_i$. Instead of requiring to manually declare such tasks, it presents a way to identify them based on the single given LTL mission specification $\mathcal{M}$. As summarized in the first part of this section, this enables the construction of a special team model with its state space linear in the number agents. We refer the interested reader to [8] for further details. Assuming this choice of a team model, in the following denoted by $\mathcal{G}$ to differentiate from the general case $\mathcal{C}$, we further improve efficiency of applying Algorithm 1 to $\mathcal{G}$.

A. Team Model Construction

The main challenge is to find all tasks $\mathcal{T}_i$ of $\mathcal{M}$. For this purpose, [8] defines a *decomposition set* $D \subseteq Q$ in the NFA $\mathcal{F}$ of $\mathcal{M}$. In order to decide which state belongs to D, an *essential sequence* σ_e is defined as the sequence $\sigma \colon \mathbb{N} \to 2^\Pi$ of required propositions $\pi \in \Pi$ such that σ_e would violate $\mathcal{M}$ if one of the propositions π would be missing. Then, a state q belongs to D if and only if there exists an accepting run in the NFA through q such that application of the essential sequence σ_e describing this run still yields an accepting run if both parts of σ_e, the one leading to q and the one after q, are swapped.

Based on the mission decomposition choices given by the decomposition set D, a team model can be constructed as follows to plan the optimal allocation of tasks across the team of agents. As illustrated in Figure 2, the product automata $\mathcal{P}_r$ of all individual agents $r \in \{1, ..., N\}$ are connected by so-called *switch transitions*.

Definition 4 (Decomposition Team Model)**.** *The automaton $\mathcal{G}$ is a union of the N local product automata $\mathcal{P}_r$ with $r \in \{1, ..., N\}$ and given by $\mathcal{G} := (S_\mathcal{G}, S_{0,\mathcal{G}}, F_\mathcal{G}, A_\mathcal{G})$ consisting of **(1)** a set of states $S_\mathcal{G} = \{(r, q, s) \colon r \in \{1, ..., N\}, (q, s) \in S_{\mathcal{P}_r}\}$, **(2)** a set of initial states $S_{0,\mathcal{G}} = \{(r, q, s) \in S_\mathcal{G} \colon r = 1\}$ **(3)** a set of final states $F_\mathcal{G} = \{(r, q, s) \in S_\mathcal{G} \colon q \in F\}$, **(4)** a set of actions $A_\mathcal{G} = \bigcup_r A_{\mathcal{P}_r} \cup \zeta$ which include switch transitions ζ as defined below.*

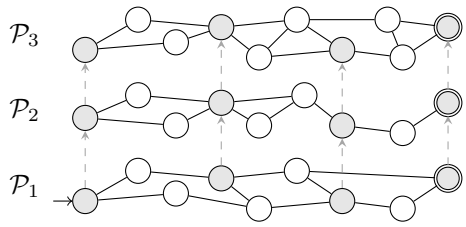

Fig. 2. Structure of $\mathcal{G}$ for three agents. It has one initial state (lower left) and three final states (right). Between the agent automata, directed *switch transitions* to the next agent connect states of the *decomposition set*.

Above, $\mathcal{P}_r$ is the product automaton corresponding to agent r according to Definition 3. A switch transition is only present at a state $s \in S_\mathcal{G}$ if $\mathcal{M}$ can be decomposed at s according to the following conditions.

Definition 5 (Switch Transition)**.** *The set of switch transitions in $\mathcal{G}$ is given by $\zeta \subset S_\mathcal{G} \times S_\mathcal{G}$. A transition $\varsigma = ((r_\mathrm{s}, q_\mathrm{s}, s_\mathrm{s}), (r_\mathrm{t}, q_\mathrm{t}, s_\mathrm{t}))$ is in ζ if and only if **(i)** connects different agents: $r_\mathrm{s} \neq r_\mathrm{t}$, **(ii)** preserves the NFA progress: $q_\mathrm{s} = q_\mathrm{t}$, **(iii)** points to the next agent: $r_\mathrm{t} = r_\mathrm{s} + 1$, **(iv)** points to an initial agent state: $s_\mathrm{t} = s_{0,\mathcal{A}}$, **(v)** represents a decomposition choice: $q_\mathrm{s} \in D$.*

Condition (v) ensures a correct decomposition of $\mathcal{M}$ in the planning process based on the results of [8], to which we refer for further details.

We can thus use the representation $\mathcal{G}$ as specific realization of the team model $\mathcal{C}$ for which Algorithm 1 is defined.

B. Planning Adaptations

Given the special structure of $\mathcal{G}$, additional labels can be eliminated in the planning process to further accelerate Algorithm 1. More specifically, we utilize the switch condition (iii) from Definition 5 to observe the following. Let r denote the agent associated with a certain state $s \in S_\mathcal{G}$. All states $v \in S_\mathcal{G}$ reachable from s can only be associated with agents $\rho_1 \geq r$, but it is not possible to consider any of the agents $\rho_2 < r$ again in the planning process. The following example illustrates a situation where it would be efficient to eliminate a presumably non-dominated label.

Example. Consider a system with four agents $\{1, 2, 3, 4\}$. Assume that the labels l and ℓ have been found by Algorithm 1 to reach a certain state s associated with agent $r = 3$ with the cost vectors $c^{(l)} = [3, 4, 1, 0]^T$ and $c^{(\ell)} = [5, 2, 1, 0]^T$. Neither $c^{(l)}$ nor $c^{(\ell)}$ dominates the other, so neither of the labels would be eliminated. However, preserving both labels is too conservative in the team model $\mathcal{G}$. For our definition of κ, we already get that $\|c^{(l)}\|_1 = \|c^{(\ell)}\|_1$ and $\|c^{(l)}\|_\infty < \|c^{(\ell)}\|_\infty$. Furthermore, only entries $c^{(l)}_{\rho_1}, c^{(\ell)}_{\rho_1}$ with $\rho_1 \geq r$ can still change. Specifically, $c^{(l)}_2 = 4$ is greater than $c^{(\ell)}_2 = 2$, but still less than $c^{(\ell)}_1 = 5$ and cannot increase anymore. Thus, it would be desirable to eliminate label ℓ. ◁

Instead of modifying Algorithm 1, we change the cost representation $c_\beta \in \mathbb{R}^N_{\geq 0}$ to a new representation $\widehat{c}_\beta \in \mathbb{R}^3_{\geq 0}$ and adapt the team cost function $\kappa\colon \mathbb{R}^N_{\geq 0} \to \mathbb{R}_{\geq 0}$ to $\widehat{\kappa}\colon \mathbb{R}^3_{\geq 0} \to \mathbb{R}_{\geq 0}$ accordingly. First, we define the three-dimensional cost vector $\widehat{c}_\beta$ such that

$$\widehat{c}_\beta = \begin{pmatrix} \|(c_{\beta,1}, \ldots, c_{\beta,r-1})^T\|_\infty \\ \|(c_{\beta,1}, \ldots, c_{\beta,r-1})^T\|_1 \\ c_{\beta,r} \end{pmatrix} \tag{5}$$

where r denotes the agent associated with the last state of β. The first two dimensions represent maximum cost and total cost of all previous agents $1, \ldots, r-1$, and the third dimension the cost of the agent r.

Lemma 4 (Cost Dominance)**.** *For all states $s \in S_\mathcal{G}$, it holds for a pair of labels l and ℓ at s that **(1)** $c^{(l)} <_\mathrm{P} c^{(\ell)}$ only if $\widehat{c}^{(l)} <_\mathrm{P} \widehat{c}^{(\ell)}$ and **(2)**, if $\widehat{c}^{(l)} <_\mathrm{P} \widehat{c}^{(\ell)}$ while $c^{(l)} \not<_\mathrm{P} c^{(\ell)}$, there exists an optimal action sequence β_opt of which construction does not include ℓ.*

Proof. Let r denote the agent to whose partition state s belongs and note that in both cases $\widehat{c}^{(l)} <_\mathrm{P} \widehat{c}^{(\ell)}$ implies that $\widehat{c}^{(l)} \neq \widehat{c}^{(\ell)}$ and consequently $c^{(l)} \neq c^{(\ell)}$. Also recall that $c^{(l)}_R, c^{(\ell)}_R = 0$ for all $R \in \{r+1, \ldots, N\}$, given by the switch condition (iii) from Definition 5.

(1) From $c^{(l)} <_\mathrm{P} c^{(\ell)}$ follows that $c^{(l)}_i \leq c^{(\ell)}_i$ for all i and there exists a j such that $c^{(l)}_j < c^{(\ell)}_j$. Consequently, we get for the components of $\widehat{c}^{(l)}$ and $\widehat{c}^{(\ell)}$ that $\widehat{c}^{(l)}_1 \leq \widehat{c}^{(\ell)}_1$, $\widehat{c}^{(l)}_2 < \widehat{c}^{(\ell)}_2$, and $\widehat{c}^{(l)}_3 = \widehat{c}^{(\ell)}_3$ for $j \neq r$ as well as $\widehat{c}^{(l)}_1 \leq \widehat{c}^{(\ell)}_1$, $\widehat{c}^{(l)}_2 \leq \widehat{c}^{(\ell)}_2$, and $\widehat{c}^{(l)}_3 < \widehat{c}^{(\ell)}_3$ for $j = r$, which shows that $\widehat{c}^{(l)} <_\mathrm{P} \widehat{c}^{(\ell)}$.

(2) For $c^{(l)} \not<_\mathrm{P} c^{(\ell)}$ and $\widehat{c}^{(l)} \neq \widehat{c}^{(\ell)}$, there needs to exist at least one k such that $c^{(l)}_k > c^{(\ell)}_k$. In addition, $\widehat{c}^{(l)} <_\mathrm{P} \widehat{c}^{(\ell)}$ implies that $k < r$. Furthermore, $\widehat{c}^{(l)}_1 \leq \widehat{c}^{(\ell)}_1$ and $\widehat{c}^{(l)}_2 \leq \widehat{c}^{(\ell)}_2$ following from $\widehat{c}^{(l)} <_\mathrm{P} \widehat{c}^{(\ell)}$ show that all components k where label l is worse than ℓ are *non-critical*. Non-critical means that sum and maximum over all agents $1, \ldots, r-1$ for l are still at least as good as for ℓ. Now consider continuations L of l and $\mathcal{L}$ of ℓ in the sense that constructing an action sequence from L includes l. Again given by the switch condition (iii), no entry $c^{(L)}_\rho, c^{(\mathcal{L})}_\rho$ with $\rho \in \{1, \ldots, r-1\}$ can change. This implies that the components where L is worse than $\mathcal{L}$ will remain non-critical. Consequently, either ℓ is not part of constructing an optimal action sequence or there is another optimal action sequence of which construction instead includes l. □

In case **(2)** of Lemma 4, ℓ is called dominated and can be eliminated in the planning process. Next, we define the adapted team cost function $\widehat{\kappa}$ such that

$$\widehat{\kappa}(\widehat{c}_\beta) = (1-\epsilon) \cdot \|(\widehat{c}_{\beta,1}, \widehat{c}_{\beta,3})^T\|_\infty + \epsilon \cdot \|(\widehat{c}_{\beta,2}, \widehat{c}_{\beta,3})^T\|_1 \tag{6}$$

with $\epsilon \in (0, 1]$. Then, we can show that $\widehat{\kappa}(\widehat{c}_\beta)$ with $\widehat{c}_\beta$ calculated from a certain c_β always equals $\kappa(c_\beta)$.

Lemma 5 (Team Cost Equivalence)**.** *For $\widehat{c}_\beta$ and $\widehat{\kappa}$ as defined above (Eq. 5 and 6), we get $\widehat{\kappa}(\widehat{c}_\beta) = \kappa(c_\beta)$ for all possible c_β on $\mathcal{G}$.*

Proof. From $c_{\beta,\rho_2} = 0$ for all $\rho_2 \in \{r+1, \ldots, N\}$ follows that $\|c_\beta\|_\infty = \|(c_{\beta,1}, \ldots, c_{\beta,r})^T\|_\infty$ and $\|c_\beta\|_1 = \|(c_{\beta,1}, \ldots, c_{\beta,r})^T\|_1$. Consequently,

$$\begin{aligned}\widehat{\kappa}(\widehat{c}_\beta) &= (1-\epsilon)\cdot\|(\widehat{c}_{\beta,1},\widehat{c}_{\beta,3})^T\|_\infty + \epsilon\cdot\|(\widehat{c}_{\beta,2},\widehat{c}_{\beta,3})^T\|_1\\ &= (1-\epsilon)\cdot\|(c_{\beta,1},\ldots,c_{\beta,r})^T\|_\infty\\ &\quad +\epsilon\cdot\|(c_{\beta,1},\ldots,c_{\beta,r})^T\|_1\\ &= (1-\epsilon)\cdot\|c_\beta\|_\infty + \epsilon\cdot\|c_\beta\|_1\\ &= \kappa(c_\beta)\end{aligned}$$

for all action sequences β to all states $s \in S_\mathcal{G}$. □

Finally, the resulting team action sequence β_{fin} over $\mathcal{G}$ for the team of agents, which is a concatenation of independent and parallel executable action sequences for all agents $r \in \{1,\ldots,N\}$, can be split to form individual action sequences $\beta^{(r)}$. Specifically, $\beta^{(r)}$ is given by the respective part of β_{fin} of which all states and intermediate actions belong to partition r, and all $\beta^{(r)}$ are separated by switch transitions. Agents for which $\beta^{(r)}$ does not contain any action are not considered as part of the team involved in the mission $\mathcal{M}$.

VI. Case Study

The presented approach has been implemented in ROS and any robot capability available in ROS can be encapsulated as an action of the agent model $\mathcal{A}$. Specifically, we use a topological map like the one shown in Figure 3 (top) to represent navigation and define further actions like *pick-up*, restricted to certain locations.

In order to execute the planned action sequences, the behavior executive *FlexBE*[1] [21] is leveraged. FlexBE lets a system provide a set of parametrizable capabilities, for example navigation to a certain waypoint. This can be annotated to the agent model $\mathcal{A}$ such that each edge in the model refers to one executable robot action. For the translation of an LTL formula to its equivalent NFA, the model checking library *Spot*[2] [22] is used.

Our implementation supports a robotic system where each robot communicates with a common base station. Each robot registers with its agent model $\mathcal{A}$ when available for a new mission. When given a new LTL specification $\mathcal{M}$, the base station plans execution for the currently available team and finally distributes the respective action sequences $\beta^{(r)}$ to all robots r which are part of the optimal solution.

A. Mission Description

Assume a multi-robot system with three robots in a hotel setting based on Figure 3. The robots start at the indicated positions in their default state, each with an agent model formed as the product of both parts of Figure 3. In the considered mission $\mathcal{M}_1$, the robots need to deliver drinks to hotel rooms h_1, h_2, h_3, h_4. In addition, robots should avoid public areas p while carrying a drink.

$$\mathcal{M}_1 = \left(\Diamond_{i\in\{1,2,3,4\}}(h_i \wedge c \wedge \circ\neg c)\right) \wedge \Box(c \implies \neg p)$$

Resulting from the constraint, robot $R3$ cannot pick up a drink at s_2, but needs to go to s_1. As a consequence, $R3$ only serves h_3, which is close to s_1. The complete result is shown in Figure 4 (left) and has the cost vector $c_\beta = (29, 26, 28)^T$.

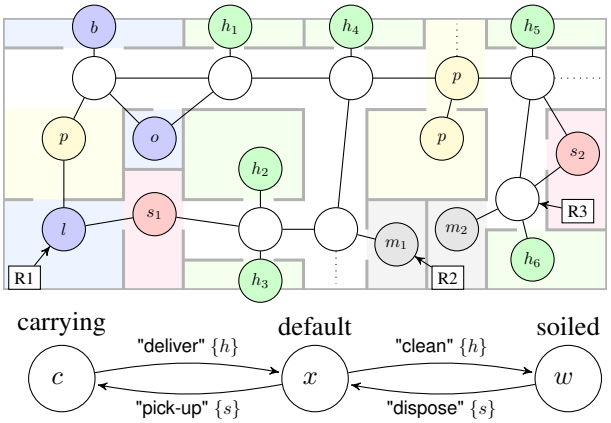

Fig. 3. **(top)** Topological map of the environment to represent navigation actions, while nodes are labeled with atomic propositions. **(bottom)** Location-independent states of a robot. Each action, modeled as a transition, has propositional conditions, for example "pick-up" is only possible at room service locations (proposition $s = \{s_1, s_2\}$).

As a variation of $\mathcal{M}_1$, we assume $\mathcal{M}_2$ with additional resource constraints as follows. Each robot has a battery level and can repeatedly recharge 5% battery at maintenance locations m_1, m_2 if not carrying a drink. Initially, $R1$ has 65% battery, $R2$ has 58%, and $R3$ has 53%. Furthermore, the public area constraint is removed and instead, the amounts of drinks at s_1 is limited to only two drinks, resulting in $\gamma_0 = (0.65, 0.58, 0.53, 2)^T$. The result is as well depicted in Figure 4 (right) with costs $c_\beta = (42, 40, 28)^T$ and final resources $\gamma_\beta = (0.05, 0.04, 0.02, 0)^T$. All three robots need to charge during the mission[3].

B. Discussion

Although there exist conceptually related approaches, e.g., [5] based on MILP, we are not aware of any other implementation that minimizes maximal execution time for a team of robots, given an LTL mission and resource constraints. Thus, the following discussion is limited to different aspects of our proposed planning approach.

Figure 5 lists experimental results regarding planning performance for the two missions and additional variations of $\mathcal{M}_2$ to illustrate the effect of resource constraints on planning performance. *Label updates* is the number iterations of the while-loop in Algorithm 1. The increased planning time for $\mathcal{M}_2$ results from the additional Pareto optimal

[1]http://flexbe.github.io [2]http://spot.lrde.epita.fr

[3]Video of simulated $\mathcal{M}_2$ and another demonstration with two real robots: https://youtu.be/Boor9kW44ko

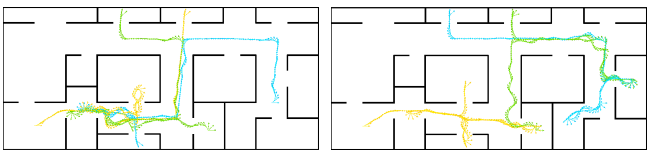

Fig. 4. Localization recordings of a simulation run, **(left)** mission $\mathcal{M}_1$ and **(right)** mission $\mathcal{M}_2$.

	t_{plan} (sec)	label updates
$\mathcal{M}_1$	0.378	2,474
$\mathcal{M}_2$	3.804	19,248
$\mathcal{M}_2^{battery}$	2.358	12,409
$\mathcal{M}_2^{drinks}$	0.428	2,999
$\mathcal{M}_2^{none}$	0.265	1,825

Fig. 5. Planning time and number of label updates for the two missions. In addition, we evaluate $\mathcal{M}_2$ with only constraining battery $\mathcal{M}_2^{battery}$, only drinks $\mathcal{M}_2^{drinks}$, and no constraints $\mathcal{M}_2^{none}$.

labels, caused by resource consumptions and recharging actions. Especially recharging, inversely proportional to costs, appears to significantly impact planning time since many more Pareto optimal choices need to be explored.

Besides the considered resources, also the complexity of the mission influences planning time. Primarily, this results from constructing $\mathcal{F}$ from $\mathcal{M}$, which has a worst-case complexity exponential in the length of the formula [3]. This is inherent from the expressiveness of LTL, e.g., consider that a traveling salesman problem can be formulated in LTL. Figure 6 depicts the planning time for $\mathcal{M}_1$ with different team sizes. Ten robots need roughly 1s planning time and one hundred robots around 40s. This scalability mainly results from the model representation and planning adaptations discussed in Section V.

VII. Conclusions

We presented an efficient approach for optimal LTL multi-robot planning based on multi-objective shortest path search in a graph representation which combines both the robot system and a finite LTL mission specification. The planning algorithm enables to consider resource constraints, as illustrated in the discussed case study, which is especially relevant for the application of multi-robot planning in mid- and long-term scenarios. Combined with previous results from the area of LTL mission decomposition, the presented algorithm plans optimal distribution of a given mission specification among a dynamically formed team of robots.

References

[1] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans. on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[2] A. Ulusoy, S. Smith, X. C. Ding, and C. Belta, "Robust multi-robot optimal path planning with temporal logic constraints," in *Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 4693–4698.

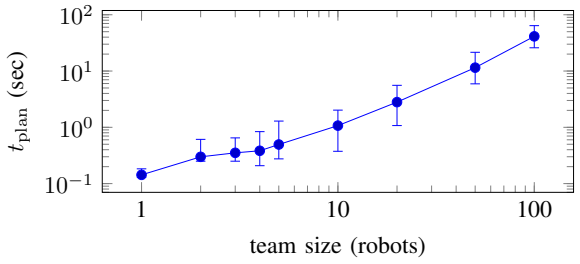

Fig. 6. Average planning time for $\mathcal{M}_1$ with respect to the team size and random initial positions. In total, more than 1,000 runs have been recorded. Error bars indicate minimal and maximal values.

[3] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[4] M. Guo and D. V. Dimarogonas, "Bottom-up motion and task coordination for loosely-coupled multi-agent systems with dependent local tasks," in *CASE*. IEEE, 2015, pp. 348–355.

[5] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-UAV mission planning," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1372–1395, 2011.

[6] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal approach to the deployment of distributed robotic teams," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 158–171, 2012.

[7] J. Tumova and D. V. Dimarogonas, "Decomposition of Multi-Agent Planning under Distributed Motion and Task LTL Specifications," in *CDC*. IEEE, 2015, pp. 1775–1780.

[8] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Decomposition of finite LTL specifications for efficient multi-agent planning," in *Int. Symp. on Distributed Autonomous Robotic Systems (DARS)*. Springer, 2016.

[9] E. M. Wolff, U. Topcu, and R. M. Murray, "Efficient reactive controller synthesis for a fragment of linear temporal logic," in *Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 5033–5040.

[10] M. Ehrgott, *Multicriteria optimization*. Springer Science & Business Media, 2006.

[11] J. M. Paixão and J. L. Santos, "Labeling Methods for the General Case of the Multi-objective Shortest Path Problem–A Computational Study," in *Computational Intelligence and Decision Making*. Springer, 2013, pp. 489–502.

[12] X. Gandibleux, F. Beugnies, and S. Randriamasy, "Martins' algorithm revisited for multi-objective shortest path problems with a MaxMin cost function," *4OR*, vol. 4, no. 1, pp. 47–59, 2006.

[13] D. P. Bertsekas, F. Guerriero, and R. Musmanno, "Parallel asynchronous label-correcting methods for shortest paths," *Journal of Optimization Theory and Applications*, vol. 88, no. 2, pp. 297–320, 1996.

[14] D. W. Pentico, "Assignment problems: A golden anniversary survey," *European Journal of Operational Research*, vol. 176, no. 2, pp. 774–793, 2007.

[15] R. E. Burkard and E. Cela, "Linear assignment problems and extensions," in *Handbook of combinatorial optimization*. Springer, 1999, pp. 75–149.

[16] E. Q. V. Martins, "On a multicriteria shortest path problem," *European Journal of Operational Research*, vol. 16, no. 2, pp. 236–245, 1984.

[17] S. Irnich and G. Desaulniers, "Shortest path problems with resource constraints," in *Column generation*. Springer, 2005, pp. 33–65.

[18] N. Boland, J. Dethridge, and I. Dumitrescu, "Accelerated label setting algorithms for the elementary resource constrained shortest path problem," *Operations Research Letters*, vol. 34, no. 1, pp. 58–68, 2006.

[19] G. De Giacomo, R. De Masellis, and M. Montali, "Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness." in *AAAI*. Citeseer, 2014, pp. 1027–1033.

[20] O. Kupferman and M. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[21] P. Schillinger, S. Kohlbrecher, and O. von Stryk, "Human-Robot Collaborative High-Level Control with Application to Rescue Robotics," in *IEEE Int. Conf. on Robotics and Automation*, Stockholm, Sweden, May 2016.

[22] 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 *ATVA*. Springer, 2016.