Spatio-Temporal Planning for a Reconfigurable Multi-Robot System

Thomas M. Roehr[1] and **Frank Kirchner**[1,2]
[1] DFKI GmbH Robotics Innovation Center, Bremen, Germany
{thomas.roehr, frank.kirchner}@dfki.de
[2] Robotics Group, Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany

Abstract

In this paper we introduce a spatio-temporal planning and scheduling approach for collaborative multi-robot systems. In particular, we are targeting an application to physically reconfigurable systems in order to take advantage of morphological changes. The planning approach relies on an ontology to model the functionalities individual physical agents offer within the multi-robot system, while an implicit domain representation is given. An inference layer on top of a knowledge-based system allows to account for superadditive effects from physically combining two or more robots. We present a formulation of the domain-specific planning problem and outline our spatio-temporal planning approach. This approach combines the use of constraint-based satisfaction techniques with linear optimization to solve a multicommodity min-cost flow problem to deal with the transportation of immobile robotic systems. The paper describes the findings after implementing core features and evaluates the approach based on a fictitious scientific mission. We close with a discussion of the current limitations of the illustrated approach.

1 Introduction

Robotic space missions rely on highly specialized robotic systems to perform scientific missions. Despite the existing capabilities of these systems, the communication delay in a space-exploratory mission has a significant influence on operations and is a major reason for an overall slow mission progress (NASA 2016). While automation and introduction of techniques from artificial intelligence could mitigate some of these effects, there is a reluctance of increasing autonomy in such systems. One of the reasons for this reluctance is keeping the operational and financial risk to a minimum.

The application of reconfigurable multi-robot systems (Roehr, Cordes, and Kirchner 2014) offers an approach to reduce these types of risk in multiple ways. Firstly, the modularity of such system allows for an incremental mission design, which can include systems with different degrees of specialization. That means, that after an initial deployment phase where a limited number of robotic systems is used, subsequent system development and deployments can take advantage of reconfigurability and extend the functionality of the existing systems. Secondly, the additional degree of freedom that arises from the flexibility of reorganising resources (Evans 1991) within the multi-robot system can be used at the time of mission planning as well as at runtime. Thus, the inherent flexibility (DeLoach and Kolesnikov 2006) of a physically reconfigurable multi-robot system offers a benefit for operations regarding safety, efficiency and efficacy. A reconfigurable multi-robot system encompasses all the benefits of a multi-robot system so that it can mitigate the issue of single-point-of-failure. As an additional benefit, resources can be dynamically shifted within a physically reconfigurable system, so that functionalities and redundancies can be created where they are needed the most. Meanwhile, efficiency can be increased by distributing tasks according to the level of capability of individual robots and while traditional multi-robot systems can increase efficacy by applying general cooperation schemas, reconfiguration adds the possibility of creating physical coalitions aside from creating virtual ones. This not only results in a morphological change, but allows to take advantage of any superadditive effect, e.g., using abilities that are not available within individual systems, but only when two or more join.

The motivation for developing a planner for reconfigurable multi-robot systems comes from the idea of taking full advantage of the available flexibility in an automated way. While will not present results of a fully completed implementation that includes the anticipated multi-objective optimization for redundancy and efficiency, but we illustrate an overall integrated planning approach that accounts for the full flexibility of a reconfigurable multi-robot system. Hence, our main contribution is the outline of an integrated multi-robot planning approach, which allows us to limit the combinatorial explosion that comes with a cooperative multi-robot system. In the following we therefore introduce related work in this area in Section 2. Subsequently, we outline our planning approach in Section 3 and provide details on the implementation and validation in Section 4. The final section 5 provides our conclusions and an outlook of future work.

2 Background

Real reconfigurable multi-robot systems have been successfully developed in the past years, though the main focus of these efforts has been swarm-like systems, i.e. modules that can be almost arbitrarily combined, but are limited with

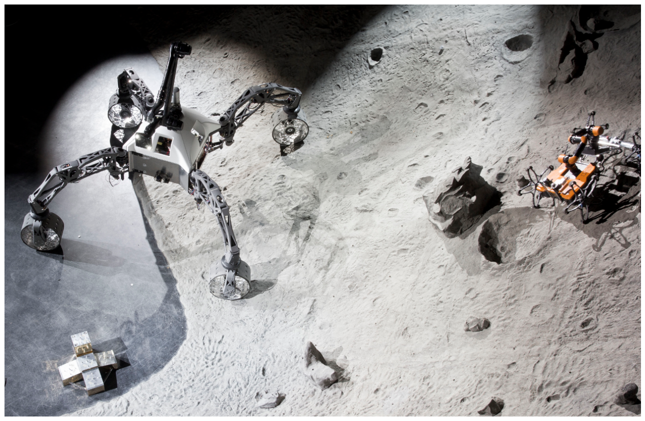


Figure 1: The multi-robot system which represents the initial motivation and target platform for the planning algorithm.

respect to their final applicability for complex tasks. The work by Roehr et al. (Roehr, Cordes, and Kirchner 2014) presents a recent achievement in this area by allowing more complex and more capable robots to combine and dynamically form coalitions. The project TransTerrA (Sonsalla et al. 2014) builds upon these experiences and develops a reconfigurable multi-robot system that comprises of mobile and immobile robotic agents such as the mobile rover SherpaTT, the mobile shuttle Coyote III, payload-items and so-called base camps; the robotic systems are depicted in Figure 1.

Leading to the progress of developing these kind of reconfigurable multi-robot systems has been the development of interfaces that allow for a reliable electro-mechanical coupling of two physical agents. While the development of such hardware is progressing, the automated exploitation of this additional degree of freedom has yet received little attention.

Theoretical work for organization modelling has been collected by Dignum (Dignum 2009) and OperA and LOA (logic for agent organizations) are two examples providing a formal foundation. Dignum also stresses the importance of dealing with reconfigurable systems as yet another dimension for organizations. Closer oriented towards the multi-agent and multi-robot domain are MOISE+ (Hübner, Sichman, and Boissier 2002) and OMACS (Zhong and DeLoach 2011), which provide models to describe goal oriented organizations to consider restructuring (of task assignments) of such multi-agent systems as error response and based on a metric, which quantifies the ability to fulfil specific tasks. In order to quantify the benefit of a specific configuration of an organization the work of DeLoach et al. (DeLoach and Kolesnikov 2006) is a rare example and relies on a static analysis at design stage.

Looking at the safety property a reconfigurable multi-robot system could be viewed as the so-called superadditive game (Weiss 2009) - a subclass of characteristic function games; the best coalition of agents is a monolith composite agent, i.e. merging all available agents leads to a maximum degree of redundancy. However, a reconfigurable multi-robot system is restricted in the way composite agents can be constructed, i.e. physical interfaces might be already used or are incompatible with each other, thus possible combinations are limited. Rahwan et al. (Rahwan et al. 2011) provide an approach to find the best possible combination for this restricted case, however, the given approach already requires knowledge about compatibility, which becomes impractical even for medium sized teams when agents can be connected via multiple interfaces. Meanwhile, approaches such as ModRED (Baca et al. 2014), and coalition structure generation (Rahwan et al. 2009),(Rahwan et al. 2011) provide means to identify (near) optimal configurations for unrestricted coalition building. Since the search space is the space of all so-called coalition structures $\mathcal{P}^A = 2^{|A|}$, where A is the set of available agents; the cost for this computation can be significant: Rahwan et al. show that it is (Rahwan et al. 2009) $O(2^{|A|})$ and limiting even for a small team size; stronger assumptions allow to reduce this restriction, e.g., Baca et al. (Baca et al. 2014) show a reduction to $O(log|A|)$ but they also assume a constant utility of two linked agents irrespective of the total size the coalition they are in.

Tenorth and Beetz (Tenorth and Beetz 2013) illustrate that ontologies offer a scalable approach to model robotic systems and plan with this information, e.g., they rely on a set of ontologies to describe actions, capabilities and interdependencies. However, they use their ontological description to parametrize so-called action-recipes and focus on single robotic systems, whereas Cashmore et al.(Cashmore et al. 2015) embed ontologies into a full planning approach.

Planning for reconfigurable multi-robot systems is challenging due to their high degree of flexibility and marsupial-like robotic teams (Murphy, Ausmus, and Bugajska 1999) are only one of multiple possible expressions of such reconfigurable multi-robot systems. Wurm et al. (Wurm et al. 2013) apply temporal planning in the context of a multi-robot carrier service which enhances robotic exploration. They use the PDDL-based temporal-planning system TFD/M (Eyerich, Mattmüller, and Röger 2012) which offers the use of so-called semantic attachments or rather external evaluation functions for better context integration, e.g., allowing to use a path-planning component to compute action costs. Similarly, Eich et al. (Eich et al. 2014) perform coordination of a multi-robot team using hierarchical task networks (HTNs), but without explicitly using temporal planning. A combination of (meta-)constraint-based solvers, temporal planning and HTN planning can be found in CHIMP (Stock et al. 2015), which has also been successfully applied to single and multi-robot problems. While all mentioned planning systems are suitable to multi-robot planning, none of these systems takes into account superadditive effects of combining two or more robots.

Numerous timeline-based approaches have been gathered for space-related applications by Chien et al. (Chien et al. 2012) and in our approach we also use a timeline-based representation and Temporal Constraint Networks (Nau, Ghallab, and Traverso 2004),(Dechter 2003) for qualitative temporal reasoning. In addition, we adopt the PLASMA planner approach (Maio et al. 2015) of resource driven planning.

3 Planning with a reconfigurable multi-robot system

The challenge of planning with reconfigurable multi-robot system is directly related to characteristic function games (Weiss 2009). A configuration of the multi-robot system can be viewed as a so-called coalition structure $CS \in \mathcal{P}^A$, where $\mathcal{P}^A$ represents the space of all coalitions structures that can be created from the set of actors A. Rahwan et al. formalize in (Weiss 2009) a coalition structure over A as $CS = \{C_1, \ldots, C_{|CS|}\}$ such that $\bigcup_{i=1}^{|CS|} C_i = A$. By definition coalitions within a coalition structure cannot overlap, i.e., $C_i \cap C_j = \emptyset, i, j = 1 \ldots |CS|, i \neq j$. That means, that each agent of A is part of one and only one coalition in a coalition structure.

Planning needs to consider possible transitions between these coalition structures, while finding the optimal coalition structure is already $O(2^{|A|})$ for a set of agents A (Rahwan et al. 2009).

The planning goal is to provide a feasible plan which outlines core actions to perform a robotic mission, i.e., the mission specification can be seen as the goal specification. The initial intention, however, it not to produce a fully detailed action plan, e.g., when and how manipulation takes place. Instead, the goal lies in exploiting reconfigurability to form composite agents while guaranteeing that resources for specific functionality (and thus actions) will be available at a specific location and time. Thus, a solution will represent a rather coarse grid for a robotic mission (but fulfilling necessary preconditions for functionalities) which can be used by more specialized planners, e.g., manipulation planner or navigation planner, to provide a detailed plan.

In the following we introduce the basic notation, definitions and our assumptions regarding reconfigurable multi-robot systems such as implemented in Roehr et al. (Roehr, Cordes, and Kirchner 2014):

Definition 3.1. *An* ***atomic agent*** $a \in A$ *represents a monolithic physical robotic system, where* $A = \{a_0, \ldots, a_n\}$ *is the set of all atomic agents. An atomic agent cannot be separated into two or more physical agents.*

Definition 3.2. *A mechanically coupled system of two or more atomic agents is denoted a* ***composite agent*** $CA = \{a_i, \ldots, a_j\}$*, where* $a_i, \ldots, a_j \in A, |A| \geq |CA| > 1$.

Definition 3.3. *The type of an atomic agent* a *is denoted* $\widehat{a}$ *and equivalently for a composite agent* CA *the type is denoted* $\widehat{CA}$*. The set of all agent types is denoted* $\widehat{A} = \{\widehat{a_1}, \ldots, \widehat{a_n}\}$.

Definition 3.4. *A (general) agent type* $\widehat{GA}$ *is represented as a tuple set of agent type and type cardinality:* $\widehat{GA} = \{(\widehat{a}_0, c_0), \ldots, (\widehat{a}_n, c_n)\}$*, where* $a_i \in A$ *and* $0 \leq c_i \leq |A|$. $\widehat{GA} \supset \widehat{GA'} \iff \forall (a_i, c_i) \in \widehat{GA}, (\widehat{a}_i, c_i') \in \widehat{GA'} : c_i > c_i'$*, where* $i = 1 \ldots |A|$*. Such a tuple set will be denoted an* ***agent pool****.*

Definition 3.5. *A* ***reconfigurable multi-robot system*** *(RMRS) is a set of fully cooperative atomic agents. It can temporarily form composite agents from two or more atomic agents.*

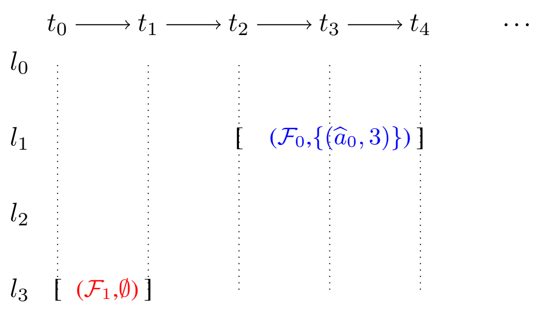

$t_0 \longrightarrow t_1 \longrightarrow t_2 \longrightarrow t_3 \longrightarrow t_4 \quad \cdots$
l_0
l_1
l_2
l_3
$[\ (\mathcal{F}_0, \{(\widehat{a}_0, 3)\})\ ]$
$[\ (\mathcal{F}_1, \emptyset)\ ]$

Figure 2: A mission specification example consisting of two spatio-temporal requirements $(\mathcal{F}_1, \emptyset)@(l_3, [t_0, t_1])$ and $(\mathcal{F}_0, \{(\widehat{a}_0, 3)\})@(l_1, [t_2, t_4])$, where $l_0, \ldots, l_3$ are location variables and $t_0, \ldots, t_4$ are timepoint variables.

Assumption 3.1. *Each agent can be mapped to a single agent type only.*

Assumption 3.2. *Each atomic and composite agent comprises a central controller.*

Assumption 3.3. *A mechanical coupling between two atomic agents can only be established through two compatible coupling interfaces.*

Formally, a **robotic mission** is a tuple $\mathcal{M} = (A_a, STR, \mathcal{X})$, where $A_a = \{a_0, \ldots, a_n\}$ is the set of available atomic agents, STR is a set of spatio-temporally qualified expressions (*steqs*) and $\mathcal{X}$ is a set of (temporal) constraints.

A spatio-temporally qualified expression in this context is an expression of the form: $(\mathcal{F}, \mathcal{A}_r)@(l, [t_s, t_e])$, where $\mathcal{F}$ is a set of functionality constants, $\mathcal{A}_r$ is a set of required (general) agent types, l is a location variable, and t_s, t_e are temporal variables describing a temporal interval with the implicit constraint $t_s < t_e$. Currently, we use qualitative timepoints and favour the notation of time slots by start and end time over the specification of a duration, since this (a) allows a future addition of quantitative timepoints (and mix between quantitative and qualitative time), and (b) can be directly translated to the problem solver for constraint-satisfaction problems (CSPs). Figure 2 illustrates a mission specification example. A mission specification can contain partially or fully temporally ordered requirements, i.e., constraints between all qualitative timepoints can remain incomplete in this specification.

Organization model

Reconfigurable multi-robot systems come with great flexibility and as already mentioned come with the possibility of increasing efficacy through cooperation. One of the challenges in planning with such a system lies in accounting for superadditive effects of composite agents, e.g., in our work we want to account for functionality that becomes only available when two or more agents join together - an example: a robot 'mobile' that is mobile, can provide power to external modules, but has no inbuilt camera, can pick up an (unpowered) camera module 'cam'. In contrast to the atomic agents ('mobile' and 'cam') the newly formed composite agent 'mobile_cam' can take images and take them from any location the robot can reach. We will later refer to this ability of this composite system as 'LocationImageProvider'.

Furthermore, in order to use planning to improve the safety of operations, we have to provide holistic metrics, i.e. we have to account for the state of the multi-robot system as a whole. The introduction of holistic metrics will allow to compare different states and characterise a multi-robot system, e.g., using the overall redundancy level with respect to the required functionality.

To model a multi-robot system, allow for inferring capabilities of agent coalitions and eventually use the model for planning we bring the ideas of Tenorth and Beetz (Tenorth and Beetz 2013) and Cashmore et al. (Cashmore et al. 2015) together. Both our requirements for planning with reconfigurable multi-robot systems, i.e. reasoning for composite agents and providing a holistic metric, are tackled by using a knowledge-base which we denote *organization model*.

The organization model is an ontology that can be augmented with domain-specific and system-specific information; it allows to encode basic functionalities of agents as well as the dependence of an agent's functionality towards the availability of other resources. We account for two resource concepts to describe functionality of an agent: capability and service. A functionality can have resource requirements which are defined using qualified cardinality constraints on the property has; this quantifies ownership of a resource. Figure 3 is an excerpt of an organization model and shows the basic modeling approach. Requirements are defined using *minimum* cardinality constraints in order to allow accounting for redundancies. Meanwhile, resources associated with an agent are defined as *maximum* cardinalities; this allows to encode resource outages for individual agents in the organization model. The organization model can further be augmented with more agent specific data, e.g., transport capacities or power consumption [1].

To identify functional dependencies of agents and infer the availability of a functionality f of a composite agent type $\widehat{CA}$ due to superadditive effects, the following problem definition can be used: searching for the assignment matrix M which maps available resources (described by a vector U that is derived from the given composite agent type $\widehat{CA}$) to required resources (represented by a vector L and derived from the given functionality):

$$L - M \cdot I = \vec{0}$$

subject to

$$u_j - \sum_{i=1}^{\dim(L)} m_{i,j} \geq 0, j = 1 \ldots \dim(U)$$

$$m_{i,j} = \begin{cases} \geq 0 & \iff c_{U,j} \sqsubseteq c_{L,i} \\ 0 & otherwise \end{cases}$$

[1]Example ontologies: `https://github.com/2maz/rmrs-pub`

$$\begin{array}{ll}
Capability & \sqsubseteq Functionality \sqsubseteq Resource \sqsubseteq \top \\
Service & \sqsubseteq Functionality \sqsubseteq Resource \sqsubseteq \top \\
MoveTo & \sqsubseteq Capability \\
ImgProvider & \sqsubseteq Service \\
MoveTo & \equiv \ \geq 1.has.Locomotion \\
 & \quad \sqcap \geq 1.has.Localization \\
 & \quad \sqcap \geq 1.has.Mapping \\
 & \quad \sqcap \geq 1.has.Power \\
ImgProvider & \equiv \ \geq 1.has.Camera \\
 & \quad \sqcap \geq 1.has.Power \\
LocImgProvider & \equiv \ \geq 1.has.ImgProvider \\
 & \quad \sqcap \geq 1.has.MoveTo \\
ARobot & \equiv Agent \\
 & \quad \sqcap \leq 1.has.Locomotion \\
 & \quad \sqcap \leq 1.has.Localization \\
 & \quad \sqcap \leq 1.has.Mapping \\
 & \quad \sqcap \leq 4.has.Camera \\
 & \quad \sqcap \leq 1.has.Power
\end{array}$$

Figure 3: Organization model excerpt of a Description Logic (DL) for an atomic agent concept $ARobot$. This illustrates an example formulation for a service named $LocImgProvider$ which reflects the functionality to provide images from specific locations.

$L = [l_1, \ldots, l_n]$ represents a vector of minimum cardinalities of required resources for the functionality f, $U = [u_1, \ldots, u_n]$ represents the vector of maximum available resources (provided by the set of agents forming the composite agent); n represents the number of different required resource classes. I is the all-ones matrix $\dim(U) \times 1$, M is the assignment matrix $\dim(L) \times \dim(U)$ with entries $m_{i,j}, i = 1 \ldots dim(L), j = 1 \ldots dim(U)$ which are restricted to 0 or positive integer values, and $c_{V,k}$ is the concept (here: resource class) belonging to entry $1 \leq k \leq dim(V)$ in a vector V. The value $m_{i,j}$ indicates how many instances of a particular resource class j are available to satisfy the request for resources of class i.

The following relationship between the existence of the functionality f and the existence of a solution to the stated problem exists:

$$ARobot \equiv 1.has.f \iff \exists M \; L - M \cdot I = \vec{0}$$

Thus, if an assignment matrix M can be found for a function f and a given agent concept, then the functionality is available for this concept.

Similarly, if the availability of a set of functionalities has to be tested, then the combined resource requirements of individual functionalities have to be considered. We define the required set of resources L_f for a functionality f as:

$$L_f = [l_{f,1}, \ldots, l_{f,n}]$$

The required resources for the set of functionalities $F = \{f_1, \ldots, f_{|F|}\}$ can then be represented as:

$$L_F = [\max(l_{f_1,1}, \ldots, l_{f_{|F|},1}), \ldots, \max(l_{f_1,n}, \ldots, l_{f_{|F|},n})]$$

At present, the organization model does not directly account for negative effects that might come with forming a composite agent; this is a limitation we intend to tackle in future revisions.

To reduce the search space we apply a functional saturation bound to a set of composite agent types $CT = \{\widehat{CA}_0, \dots, \widehat{CA}_n\}$ which provides a minimal set of coalition types $CT_{min} \subseteq CT$ that fulfil the functional requirements, so that for any other coalition type $\widehat{CA}_v \notin CT_{min}$ the following holds:

$$\exists C \in CT_{min} : \widehat{CA}_v \supset C$$

To give a more intuitive interpretation of this formulation: we intend to remove any composite agent type from the search space when a subset of its embodied agents is sufficient to provide a requested functionality. We encode this problem as integer-linear program and solve it with a standard solver (cf. Section 4) once for a given mission specification, i.e. initially when required functionalities and the available agent pool are known.

Metrics and Heuristics

To estimate travel cost for an agent we take the same approach as Wurm et al. (Wurm et al. 2013) and estimate the travel time between two locations; we define a nominal speed as a default property for mobile agents, so that based on this information the duration estimate can be computed,

A mapping from location symbols to actual coordinates can be added to the mission specification and as long as no better information is available, e.g., from a path planner (Wurm et al. 2013), the distance between locations will be the basis for cost computations. Robotic systems consume electrical energy and thus all agents come with a nominal power consumption. Hence, duration of actions is not selected directly as cost measure, but total energy consumption. Although the power consumption can vary over time with the type of activity, we assume a constant consumption and leave this improvement as future enhancement. Furthermore, we assume (though not true in all cases) that reducing energy consumption will be a primary goal for optimizing multi-robot plans since it can be mapped to the efficiency of the mission plan.

As mentioned in the introduction, safety is another criteria that has to be taken into consideration. Reconfigurable multi-robot systems can take advantage of their flexibility to exchange resources (by forming new coalitions) in order to adjust the level of redundancy in a composite agents. This leaves the options to deliberately increase redundancy for prioritized tasks or maintain a minimum level of redundancy in general. Demanding a high level of redundancy in a mission tends towards a monolithic system that performs all tasks, while a low level of redundancy thrives for a maximum number of parallel tasks, i.e. maximising efficiency. This means that in the optimization function of the planner a (user-defined) balance has to be established to trade-off redundancy and efficiency.

We will base our safety heuristic on the standard modelling of parallel and serial component-based systems. Each component will be associated with a probability of survival, leading to an overall measure of probability of survival. This information can only be extracted by measuring the performance of the real systems, and the relation to individual components. The reliability R_f of a functionality f can be computed by accounting for parallel components, i.e., resources that are not strictly required but which can serve as replacement:

$$R_f(t) = \begin{cases} 1 - \prod_{i=1}^{n}(1 - p_i(t)) & \text{parallel system} \\ \prod_{i=1}^{n} p_i(t) & \text{serial system} \end{cases}$$

where $p_i(t)$ is the time-dependant probability of survival with $0 \leq p_i(t) \leq 1$, e.g. influenced by component degrading.

Overall, the metric can be seen as characteristic function of a characteristic function game (Weiss 2009). However, here it is a multi-objective optimization function to trade-off safety and efficiency, and the problem does not fit into any of the existing four major subclasses of monotone, superadditive, convex or simple games.

The integration of metrics remains work in progress, however, computing the reliability of a plan will based on a critical path analysis and tracing the dependency of individual functionalities on specific agents.

Planning algorithm

A dedicated domain definition is not provided to the planner, yet a planning domain is implicitly given by using the organization model and accounting for recombination of agents.

A full domain definition can be provided, e.g., by translating the organization model and inference results into Planning-Domain Definition Language (PDDL). Core actions considered for such an encoding are (cf. Figure 4): (i) move, (ii) join, and (iii) split. The *move* action represents a typical change of location and requires three parameters: agent, start location and target location, while split and join refer to a composite agent instance which is uniquely identifiable by the combination of agents.

The required set of predicates: (i) *atomic(a)*: an agent a is atomic, (ii) *operative(a)*: a composite agent a is currently assembled or an atomic agent a is operative (and thus not part of any composite agent), (iii) *at(a,l)*: an agent a is at location l, (iv) *embodies(c,a)*: a composite agent c embodies an atomic agent a, (v) *mobile(a)*: an agent a is mobile (can move by itself), and (vi) *provides(a,f)*: an agent a provides a functionality f.

The action models in Figure 4 have been tested with and transcoded from a PDDL-based representation and to facilitate the transcoding into PDDL, we initially prohibit mixing of composite and atomic agents in a new composite agent. However, this does not limit the modelling capabilities: instead we assume a complete separation of a composite agent into atomic agents when a reconfiguration takes place to form a new composite agent. The corresponding transition can be easily optimized before performing it with the real robots. Furthermore, an atomic agent is either embodied by a composite agent (and becomes a virtual instance which cannot be directly associated with a location), or it is operative and physically present at a location.

$moveto(a, l_s, l_t)$ – move actor a from start l_s to target l_t

$$\begin{array}{ll} precond: & mobile(a) \wedge operative(a) \wedge at(l_s) \wedge \neg at(l_t) \\ effects: & at(l_t) \end{array}$$

$join(c, l)$ – construct the composite actor c at location l

$$\begin{array}{ll} precond: & \forall z \in A: \neg atomic(c) \wedge \neg operative(c) \wedge \\ & ((embodies(c,z) \wedge operative(z) \wedge at(z,l)) \\ & \vee (\neg embodies(c,z)) \\ effect: & \forall z \in A: at(c,l) \wedge operative(c) \wedge \\ & (\neg embodies(c,z) \vee (\neg at(z,l) \wedge \neg operative(z)) \end{array}$$

$split(c, l)$ – split the composite actor c at location l

$$\begin{array}{ll} precond: & operative(c) \wedge at(c,l) \\ effect: & \forall z \in A: \neg operative(c) \wedge \neg at(c,l) \wedge \\ & (\neg embodies(c,z) \vee (operative(z) \wedge at(z,l)) \end{array}$$

Figure 4: Operations as part of the domain definition, for a set of atomic agent A and location variables l, l_s, l_t

While this approach offers the possibility to reuse existing PDDL-based planners in a similar as done by Wurm et al. (Wurm et al. 2013) and Cashmore et. al (Cashmore et al. 2015), we found the need for translating the organization model into this intermediate representation restrictive and counter-productive for our problem. Therefore, we employ a planning and scheduling approach that uses the organization model as an integral part. The algorithm consists of the following main steps: (1) generation of a fully specified qualitative temporal constraint network (2) typing for satisficing assignment also referred to as model assignment (3) role assignment, (4) timeline construction, (5) time-expanded network construction (6) flow optimization, (7) solution evaluation, and eventually (8) generation of multi-robot plan(s).

The core structure of the planning algorithm is illustrated in Algorithm 1, i.e. up to the flow optimization step. This variant is a simple high-level search strategy and illustrates the main ideas of this paper to tackle planning with a reconfigurable multi-robot system.

Temporal constraint network The planning algorithm starts by taking all temporal constraints of the mission and generating a temporal constraint network. Based on this input $nextQualTCN$ in Algorithm 1 computes a qualitative temporal constraint network which is consistent and has no timeline gaps – such a temporal constraint network eventually contains one constraint out of $>, <, =$ between any two qualitative timepoints.

Model assigment A least-commitment principle is applied as part of the model assignment process in order to reduce the search space of coalition structures $\mathcal{P}^A$; this is done by limiting the search to composite agents that satisfy the functional requirements for each spatio-temporal expression while ignoring supersets of such composite agents. Time overlapping requirements with the same location will be merged into one requirement.

The model assignment process requires the quantification

Algorithm 1: TemPl Version 0.1

```
Data: $\mathcal{M}$: mission spec, $minNumS$: min number of
      solutions
Result: timeline-based solutions
1  begin
2      $S = \emptyset$;
       // model assignment conflict resolvers
3      $MCR = \emptyset$;
       // role assignment conflict resolvers
4      $RCR = \emptyset$;
5      while tcn = nextQualTCN($\mathcal{M}$) do
6          while nextModelAssignment($\mathcal{M}$, tcn) do
7              while ra = nextRoleAssignment($\mathcal{M}$, ma) do
8                  rt = computeRoleTimelines($\mathcal{M}$, ra) ;
9                  ten = computeTempExpNetwork($\mathcal{M}$,
                   rt);
10                 mcf = computeMinCostFlow($\mathcal{M}$, ten);
11                 if conflictFree(mcf) then
12                     s = renderSolution($\mathcal{M}$, mcf);
13                     $S = S \cup s$;
14                     if $|S| \geq minNumS$ then
15                         return S
16                 else
17                     $RCR$ =getRAConflictResolvers(mcf);
                       $MCR$ =getMAConflictResolvers(mcf);
                       if $RCR \neq \emptyset$ then
18                         r = popResolver($RCR$);
19                         applyResolver($\mathcal{M}$, r);
20                         goto 7;
21             if $MCR \neq \emptyset$ then
                   // no role assignment found
22                 r = popResolver($MCR$);
23                 applyResolver($\mathcal{M}$, r);
24                 goto 6;
25     return S
```

of *support* of a functionality for an atomic agent type $\widehat{a}$ with respect to a resource class c:

$$support(\widehat{a}, c, f) = \frac{card_{max}(c, \widehat{a})}{card_{min}(c, f)}$$

The functions $card_{min}$ and $card_{max}$ return the minimum and maximum required cardinality for an instance of a resource class, leading to the following definition of support of a function f with respect to a resource class c:

$$support(\widehat{a}, c, f) = \begin{cases} 0 & \text{no support} \\ \geq 1 & \text{full support} \\ > 0 \text{ and } < 1 & \text{partial support} \end{cases}$$

We define the *functional saturation bound* for an atomic agent type $\widehat{a}$ with respect to functionality f using the inverse of *support*:

$$FSB(\widehat{a}, f) = \max_{c \in \mathcal{C}} \frac{1}{support(\widehat{a}, c, f)},$$

where $\mathcal{C}$ is a set of resource classes and $\forall c \in \mathcal{C}$: $card_{min}(c, f) \geq 1$ to account only for relevant resource classes. Similarly, the bound for a set of functions $\mathcal{F}$ is defined as:

$$FSB(\widehat{a}, \mathcal{F}) = \max_{f \in \mathcal{F}} FSB(\widehat{a}, f)$$

Two main interpretations of the functional saturation bound exist. First, it is a lower bound on the number of required instances of an atomic agent type to achieve a functionality (if these instances are the sole contributors). Second, it is also an upper bound for the number of instances of an atomic agent type which are actually contributing to achieve this functionality; any excess availability of this agent type is not strictly necessary, but will increase the level of resource redundancy. Hence, the functional saturation bound defines the boundary between supporting functionality and introducing redundancy.

Applying the functional saturation bound allows to reduce the number of agent types that needs to be considered for a satisficing assignment, which is subsequently solved as a CSP. Each variable in this CSP corresponds to a spatio-temporal expression defined in a mission $\mathcal{M}$; each spatio-temporal expression represents a joint requirement of functionality that needs to be fulfilled by an agent and agent (type) availability. Hence, each CSP-variable has a finite domain $D = \{\widehat{CA}_k, \ldots, \widehat{CA}_l\}$ consisting of composite agent types. The solution of this CSP contains the minimum assignments of agent types for each spatio-temporal expression.

Role assignment While the model assignment leaves unsolved what agent (instance) has to be assigned to a requirement, this will be detailed by the subsequent role assignment step. This allows to deal with concurrent activities. A role $role = (i, \widehat{a})$, where $0 < i < N$ is a tuple that represents an identifiable instance of an agent type; N is the maximum number of available agents of the type $\widehat{a}$. The role assignment process takes into account the number of available agents (per type) and introduces unification constraints to allow only feasible concurrent activities. Again, this role assignment is solved as a CSP, where the domain of a variable is the set of all available roles, although this domain will be further limited for individual requirements based on the required number and type of agents. Furthermore, all roles associated with concurrent requirements have to be distinct; this is enforced by introducing inequality constraints between time overlapping requirements. A solution to the role assignment is either empty or contains (correct-by-construction) conflict-free assignments of agent roles for each spatio-temporal expression, i.e. an agent role is associated with one location at a time only.

Having conflict free role assignments is a necessary prerequisite to produce timelines in a subsequent step, i.e. after the role assignment or unification process a timeline is computed for each role (cf. Figure 5).

Logistic network At this stage of the processing it has not been considered, whether an agent is mobile and able to change the location by its own means. Hence, for the next planning step, the algorithm starts to distinguish between mobile and immobile agents. The sub-problem of dealing with mobile and immobile systems resembles the

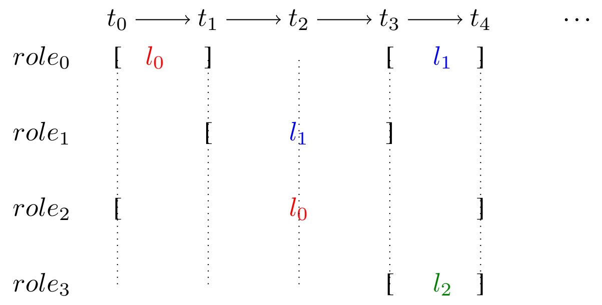


Figure 5: Timelines of multiple roles, representing the spatio-temporal requirement for each role; a role can be mapped to an instance of an agent type.

one in (Wurm et al. 2013), but can be interpreted as network flow problem (Ahuja, Magnanti, and Orlin 1993) or more specifically as a so-called *transshipment* problem.

Mobile agents can offer transport capacities, which can be used to transfer immobile agents; to distinguish between mobile and immobile systems the process relies on checking the availability of two (domain) specific functionalities: (a) 'MoveTo' describes a systems ability to perform location changes by its own means, and (b) 'TransportService' describes a system's ability to carry payload and this capability can be further detailed by specification of transport capacities. The information about available functionalities will also be defined in the organization model for each robotic agent type and clearly, this is a domain specific addition, but we are dealing with a planning problem of physical agents.

Based on the information about mobile and immobile agents, a logistic network can be modelled and solved using flow optimization techniques. We model the flow optimization problem using a time-expanded network. The time-expanded network is a directed graph $G(V, E)$. Each vertex $v \in V$ represents a tuple (l, t), where l is a location variable and t represents a (qualitative) timepoint. Each directed edge $e \in E$, $e = (v, v')$ with $v = (l, t)$ and $v' = (l', t')$ has to fulfil the temporal constraint: $t < t'$, and each role's timeline corresponds to a path in this graph (under the mentioned assumption of an underlying strongly connected temporal network). The multicommodity min-cost flow problem has been formulated as integer programming problem based on the most-general formulation with commodity dependant upper and lower bounds on edges (Kennington 1978); a commodity represents a resource type that can be transported across an edge in the graph. Each edge e has an upper bound for the overall capacity ub_e and a lower and upper bound for each commodity k: $0 \leq lb_e^k \leq ub_e^k \leq ub_e$. We extend this formulation to provide some control on the flow and allow to define a *trans-flow* constraint for a commodity on a vertex, i.e. setting a minimum inflow for a vertex that has no supply or demand for this commodity. Due to balancing constraints, i.e. inflow and outflow need to match supply or demand a valid solution will contain a symmetric outflow of the commodity from the given node.

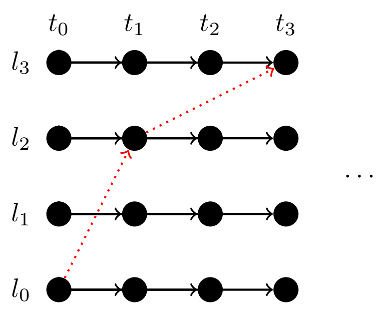


Figure 6: The initial time-expanded network with temporal constraints: $t_0 < t_1 < t_2 < t_3$ and upper capacity bound of $\leq \infty$. The dotted path represents the timeline-based path of a mobile agent a, for which an upper capacity bound UB_a can be set - given the transport capacity of this mobile agent.

In order to perform flow optimization the mission specification has to contain information about the initial location of all resources. The initial location can be interpreted as *supply* node, whereas subsequently the final destination is defined as *demand* node. There might be even unused resources that remain at the initial location as part of the solution. In order to consider the timeline for immobile systems the corresponding spatio-temporal requirements are expressed by setting supply on the start vertex, the demand on the final vertex, and the trans-flow constraint for the role, i.e. commodity it maps to, on all intermediate vertices.

Flow violations Finding a valid minimum cost flow might fail and the failed state of the min cost flow graph can be analysed to identify flow violations. Two types of violation are possible: (i) *trans-flow violation*: a commodity is not routed through a location at a specific timepoint, or (ii) *min-flow violation*: a commodity is not supplied to a location at a specific timepoint.

Both violation types can be addressed by the following resolution strategies: (a) *role distinction*: increasing the role distinction (for the amount of the missing resources) between two spatio-temporal constraints where the violation is found, or if the system is immobile (b) *transport request*: requesting the presence of a mobile system with transport capability

Plan rendering If a solution to the flow optimization problem has been found, it can be translated into a plan for the multi-robot system (or a plan per role). This solution can be characterised regarding safety and efficiency, e.g., computing the associated level of redundancy and the expected energy required to execute this solution.

4 Implementation and validation

In the following, we detail the implementation of our planning and scheduling system and while we aim at a scalable approach our application targets medium-sized reconfigurable robotic teams consisting of about 25 member agents.

We target an application on small board computers and thus try to maintain a consistent C++-based code-basis, e.g., for the ontology based organization modelling we created a C++-variant of owlapi (Horridge and Bechhofer 2011). This allows us to take advantage of ontology modelling capabilities including support for datatypes and qualified cardinality constraints. Reasoning on ontologies is based on the Description Logic (DL) reasoner FACT++ (Tsarkov and Horrocks 2006). A custom reasoner is introduced for the organization model (cf. Section 3) and relies on solving integer-linear programs. For dealing with constraint-satisfaction problems we rely on Gecode (Schulte and Tack 2012). All our graph-related subproblems such as temporal constraint networks and min-cost flow optimization share the same graph library implementation.

The specification of temporal constraints in a mission is based on qualitative relations and despite its more limited expressiveness point algebra (PA) has been selected as main representation due to better computational characteristics. Qualitative reasoning and generation of qualitative temporal constraint networks without time gaps relies on GQR (Gantner, Westphal, and Wölfl 2008) which is combined with a custom implementation of qualitative reasoning to facilitate the future combination of qualitative with quantitative temporal reasoning (Meiri 1996).

The mission specification is given by a user in XML [2] and allows to associate location constants either with tuples of latitude-longitude or Cartesian 3D coordinates; this allows the computation of metric distances between two locations (an additional radius parameter allows to map latitude-longitude coordinates to metric distances) to identify overall plan cost.

Tests have been performed on an PC equipped with an Intel CORE i7-4600U 2.1 GHz with 12 GB of memory.

A mission example

To illustrate the working of the core features of our approach we present here two example missions that are driven by some scientific goals. The following section describes one mission example using a more abstracted representation since it allows a more compact illustration. Subsequently, we will provide a more concrete and slightly more complex example, including the corresponding solution graph.

Abstracted mission We assume an initially given set of robots implicitly given by a composite actor type $\widehat{A}_I$; all agents are available to perform a mission illustrated in Figure 7. Agents of types $\widehat{a}, \widehat{b}$, and $\widehat{c}$ are mobile, where $\widehat{a}$ has a transport capacity for eight immobile systems and the remaining for one immobile system; agents of type $\widehat{d}$ and $\widehat{e}$ are immobile and as such require to join with any of the mobile agents to change their location. All or a subset of these robots can be used to produce a satisficing solution.

We assume a space-exploration mission, where all resources are initially deployed at a landing site denoted l_0. Starting from this landing site a set of tasks has to be carried

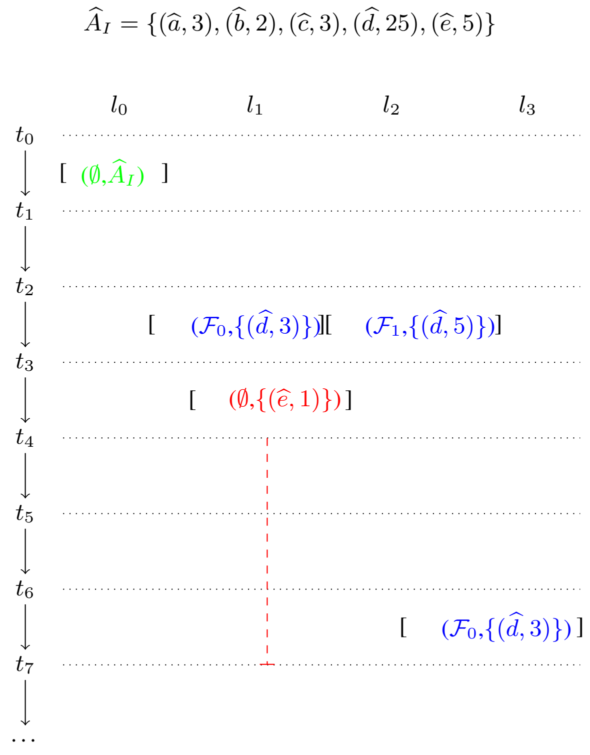
[2]Mission scenarios: `https://github.com/2maz/rmrs-pub`

out at three further locations which are denoted l_1, l_2, and $l3$. The mission specification comprises 9 locations symbols and 14 qualitative timepoints in total. The functionalities required to perform these tasks are $\mathcal{F}_0 = \{f_0, f_1\}$ and $\mathcal{F}_1 = \{f_0, f_1, f_2\}$. Functionality f_2 is only supported by immobile agents of type $\widehat{e}$. The organization model encodes the information on (superadditive) capabilities and the functional saturation bound is applied once before the planning step, i.e. leading to a mapping between composite agent types and available functionalities based on the existing set of agents.

The first iteration of the planning algorithm will lead to an incomplete solution, since the flow optimization step cannot satisfy all constraints. The problem arises through the use of a single agent of type $\widehat{e}$; after the role distribution step has taken place, a single timeline is generated for an agent of type $\widehat{e}$ which supports the functionality $(\mathcal{F}_1, \{(\widehat{d}, 5)\})$ and $(\emptyset, \widehat{A}_I)$. Initially the role distribution only takes temporal constraints into account, hence both requirements can be served by a single agent. Therefore, the flow optimization suffers from a trans-flow constraint violation on $(\mathcal{F}_1, \{(\widehat{d}, 5)\})$ after the first iteration, i.e. the required inflow and outflow for a commodity is not fulfilled. To refine the partial plan a resolver introduces additional constraints for role assignment or model distribution, e.g., to fix a role assignment the respective timeline is split, by increasing the distinction of agents between the two affected spatio-temporal requirements. If that repair action does not lead to a feasible plan, resolution has to backtrack to the level of model distribution and add a functional requirement for a mobile system.

For this particular mission example the mincommodity flow optimization requires to encode a linear problem of the following size: 4643 rows, 4290 columns and 12870 nonzeros. Columns correspond to the number of edges times the number of commodities. The problem instance is solved in about $4\,s$ including fixing one flaw in the plan. Applying the linear optimization problem does not result in a primal feasible solution and checking the first solution for constraint violations on trans-flow constraints lead to the application of a resolver. The second iteration still has no primal feasible solution, though there will be no violations on trans-flow constraints. This is due to the fact, that the excess resources exist at the starting point, i.e. a solution might leave resources unused, so that the total demand supply balance constraint will be violated. The degree of this violation can also be quantified using the internal solver results, e.g. Figure 8 illustrates output of the internally used linear program solver; the cost for routing one commodity across an edge is uniformly set to 1. Hence, the result shows that finally 28 commodities can be moved between edges while 20 resources overall remain at their initial position, i.e. are unused. The final assignment is illustrated in Figure 9, where the n^{th} instance of an agent type $\widehat{a}$ will be denoted $i_n^{\widehat{a}}$.

Eventually, splitting and joining of agent groups can be mapped to the implicit actions *join* and *split* (cf. Figure 4), while transitions between different locations map to the *move* action. These implicit action center around a set of



$\widehat{A}_I = \{(\widehat{a}, 3), (\widehat{b}, 2), (\widehat{c}, 3), (\widehat{d}, 25), (\widehat{e}, 5)\}$
l_0
l_1
l_2
l_3
t_0
t_1
t_2
t_3
t_4
t_5
t_6
t_7
…
[$(\emptyset, \widehat{A}_I)$]
[$(\mathcal{F}_0, \{(\widehat{d}, 3)\})$][$(\mathcal{F}_1, \{(\widehat{d}, 5)\})$]
[$(\emptyset, \{(\widehat{e}, 1)\})$]
[$(\mathcal{F}_0, \{(\widehat{d}, 3)\})$]

Figure 7: Mission example to describe a robotic mission: $\widehat{A}_I$ is the set of initially available resources, and the first requirement is making all resources available at location $l0$, i.e. setting the starting point. $\mathcal{F}_0 = \{f_0, f_1\}$ is a set of functionalities that needs to be available for a limited time interval.

```
GLPK Simplex Optimizer, v4.52
4643 rows, 4290 columns, 12870 non-zeros
    0: obj =     0.0  infeas =  39.0 (4500)
  500: obj =    26.0  infeas =  21.0 (4000)
 1000: obj =    26.0  infeas =  21.0 (3500)
 1356: obj =    26.0  infeas =  21.0 (3144)
LP HAS NO PRIMAL FEASIBLE SOLUTION

GLPK Simplex Optimizer, v4.52
4643 rows, 4290 columns, 12870 non-zeros
    0: obj =     0.0  infeas =  40 (4500)
  500: obj =    28.0  infeas =  20 (4000)
 1000: obj =    28.0  infeas =  20 (3500)
 1356: obj =    28.0  infeas =  20 (3144)
LP HAS NO PRIMAL FEASIBLE SOLUTION
```

Figure 8: Output of the GLPK simplex optimization (for formatting reasons we have manually shortened this output) for two subsequent planning steps: the first optimization results in an invalid solution where on 26 out of 39 commodities can be moved and triggering repairing the initial flaw by requesting the addition of another mobile agent; in the following iteration 28 out of 40 commodities be moved.

$$A = \{i_0^{\widehat{a}}, \dots, i_2^{\widehat{a}}, i_0^{\widehat{b}}, \dots i_1^{\widehat{b}}, i_0^{\widehat{c}}, \dots, i_2^{\widehat{c}}, i_0^{\widehat{d}}, \dots, i_{24}^{\widehat{d}}, i_0^{\widehat{e}}, \dots, i_4^{\widehat{e}}\}$$

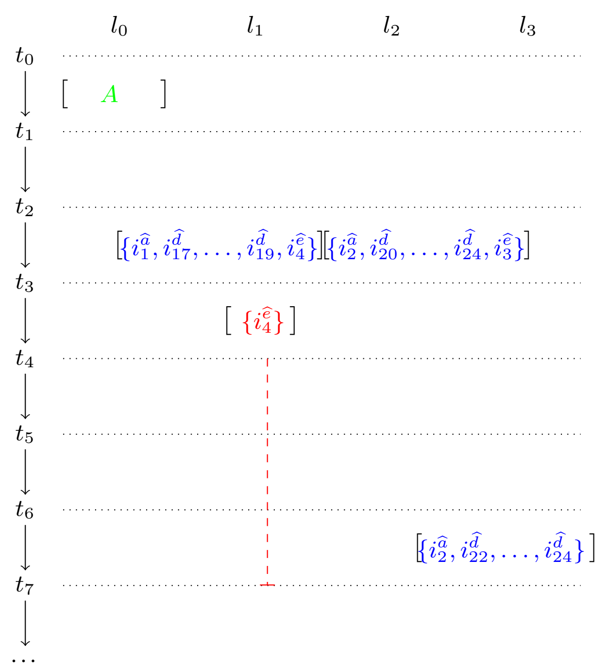

l_0
l_1
l_2
l_3
t_0
t_1
t_2
t_3
t_4
t_5
t_6
t_7
$\dots$
$[\ A\ ]$
$[\{i_1^{\widehat{a}}, i_{17}^{\widehat{d}}, \dots, i_{19}^{\widehat{d}}, i_4^{\widehat{e}}\}][\{i_2^{\widehat{a}}, i_{20}^{\widehat{d}}, \dots, i_{24}^{\widehat{d}}, i_3^{\widehat{e}}\}]$
$[\ \{i_4^{\widehat{e}}\}\ ]$
$[\{i_2^{\widehat{a}}, i_{22}^{\widehat{d}}, \dots, i_{24}^{\widehat{d}}\}\ ]$

Figure 9: Computed solution for the mission example. The initial set of agents is split into three groups, one remaining at the initial position l_0, and two groups moving to $(l_1, [t2, t3])$ and $(l_2, [t2, t3])$ respectively. An additional split allows a subgroup to continue to $l3$.

atomic agents, which can be combined to form composite agents.

Concrete mission For the concrete mission we assume that all robotic systems are initially available at the location 'lander' and a mission designer outlines the requirement based on activities that might have to be performed at certain location and in some kind of general order. The set of robot types available for this mission is based on the real systems available in (Roehr, Cordes, and Kirchner 2014) and (Sonsalla et al. 2014): the exploration rover Sherpa, the legged crater explorer CREX, the star-wheeled scout Coyote III, 25 Payloads, and 5 BaseCamps - which can be used as logistic hubs to store payloads. In this mission only a Sherpa can carry payloads and up to 8 of them. Payloads and BaseCamps are immobile units, while all other systems are mobile. The following locations are defined: $lander, b1, \dots, b7$ and qualitative timepoints $t1 < t2 < \dots < t14$.

The following spatio-temporal qualified expressions define the requirements for the mission:

1. ({},{(Sherpa,3),(CREX,2),(CoyoteIII,3),(Payload,25),(BaseCamp,5)} @(lander,[t0,t1])
2. ({},{(Payload,3})@(lander,[t4,t10])
3. ({LocationImageProvider, EmiPowerProvider},{(Payload,3)})@(b1,[t2,t3])
4. ({},{(Payload,1)})@(b1,[t3,t14])
5. ({},{(BaseCamp,1)})@(b1,[t4,t7])
6. ({LogisticHub, LocationImageProvider, EmiPowerProvider},{(Payload,3)}) @(b2,[t2,t3])
7. ({LocationImageProvider, EmiPowerProvider},{(Payload,6)}) @(b4,[t6,t7])
8. ({},{(Payload,3})@(b4,[t8,t9])
9. ({},{(BaseCamp,3)})@(b4,[t11,t14])
10. ({},{(Payload,1)})@(b6,[t10,t14])

The functionalities *LocationImageProvider* are available on Sherpa, while *EmiPowerProvider* is available on all mobile systems. On a single core a solution is computed in 57.56±9.8 s (averaged over 10 runs), a solution is illustrated in Figure 10; the linear problem to solve the transshipment problem has the following size: 9100 rows, 4320 columns and 21536 nonzeros.

Limitations This paper illustrates first results of a prototype of the planning approach outlined in the previous sections. As such, it requires further improvement and assessment to analyse computational properties and completeness. The optimization for performance has so far focused on the model-based planning approach and the use of the functional saturation bound and performance is expected to be improved further.

A detailed quantification of functionalities is not yet part of the modelling, i.e. a transport capacity can be requested in general, but not the transport capacity of a certain number of some agent type. Currently, this has to be solved by introducing additional spatio-temporal requirements which are nearby to the original one in time and space.

5 Conclusions and Future Work

This paper illustrates an approach towards automated planning and scheduling for reconfigurable multi-robot systems that accounts for the embodiment of agents and reconfigurability. In this work we bring knowledge engineering and temporal planning together to find a practical and scalable solution of dealing with reconfigurable multi-robot systems.

The organization model is a key element to allow reasoning with capabilities of atomic and composite agents. Since the organization model is encoded in Web Ontology Language (OWL), which is a specification of the World Wide Web Consortium (W3C), it offers a well defined interface to users for modelling as well as for interoperation and exchange of information within the multi-robot system. By this and similar to reconfigurable multi-robot systems it supports incremental mission design since it can grow with the system, e.g., by introducing new functionalities and agent descriptions.

Our planning approach uses the newly introduced functional saturation bound to limit the effects of combinatorial explosion. While the functional saturation bound cannot prevent combinatorial explosion, it can reduce the planning problem even when hundreds of agents are available for the mission since handling of redundant coalitions of agents is avoided. The core planning approach relies on temporal networks that have no timeline gaps, but we use a CSP-based

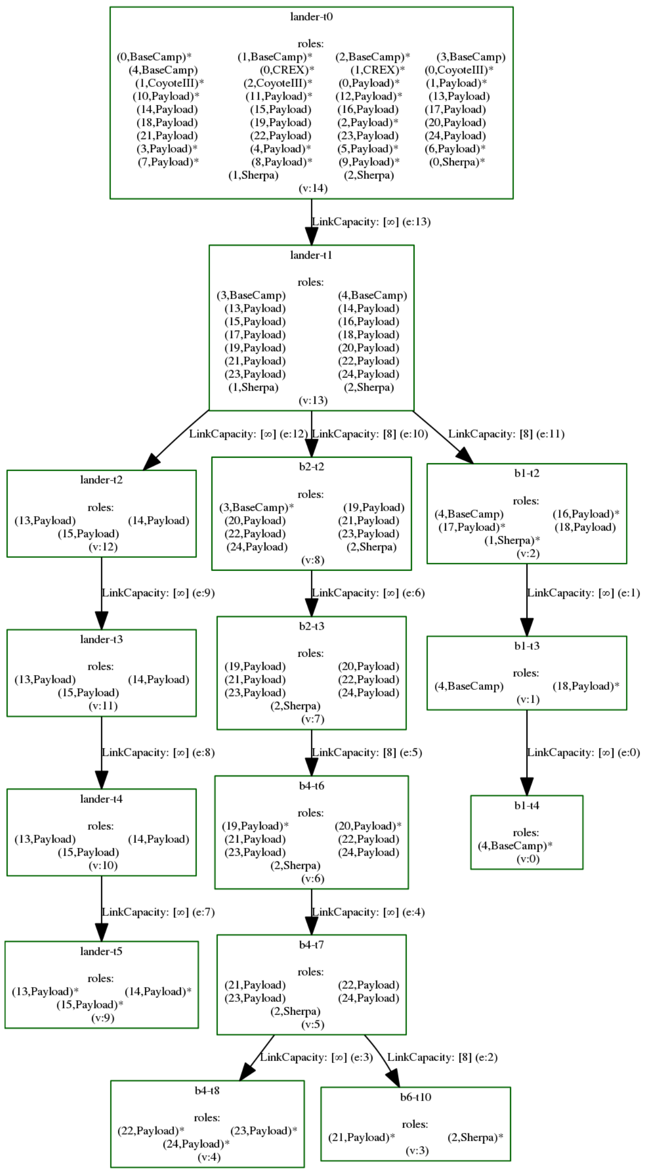

lander-t0
roles:
(0,BaseCamp)* (1,BaseCamp)* (2,BaseCamp)* (3,BaseCamp)
(4,BaseCamp) (0,CREX)* (1,CREX)* (0,CoyoteIII)*
(1,CoyoteIII)* (2,CoyoteIII)* (0,Payload)* (1,Payload)*
(10,Payload)* (11,Payload)* (12,Payload)* (13,Payload)
(14,Payload) (15,Payload) (16,Payload) (17,Payload)
(18,Payload) (19,Payload) (2,Payload)* (20,Payload)
(21,Payload) (22,Payload) (23,Payload) (24,Payload)
(3,Payload)* (4,Payload)* (5,Payload)* (6,Payload)*
(7,Payload)* (8,Payload)* (9,Payload)* (0,Sherpa)*
(1,Sherpa) (2,Sherpa)
(v:14)
LinkCapacity: [∞] (e:13)
lander-t1
roles:
(3,BaseCamp) (4,BaseCamp)
(13,Payload) (14,Payload)
(15,Payload) (16,Payload)
(17,Payload) (18,Payload)
(19,Payload) (20,Payload)
(21,Payload) (22,Payload)
(23,Payload) (24,Payload)
(1,Sherpa) (2,Sherpa)
(v:13)
LinkCapacity: [∞] (e:12)
LinkCapacity: [8] (e:10)
LinkCapacity: [8] (e:11)
lander-t2
roles:
(13,Payload) (14,Payload)
(15,Payload)
(v:12)
b2-t2
roles:
(3,BaseCamp)* (19,Payload)
(20,Payload) (21,Payload)
(22,Payload) (23,Payload)
(24,Payload) (2,Sherpa)
(v:8)
b1-t2
roles:
(4,BaseCamp) (16,Payload)*
(17,Payload)* (18,Payload)
(1,Sherpa)*
(v:2)
LinkCapacity: [∞] (e:9)
LinkCapacity: [∞] (e:6)
LinkCapacity: [∞] (e:1)
lander-t3
roles:
(13,Payload) (14,Payload)
(15,Payload)
(v:11)
b2-t3
roles:
(19,Payload) (20,Payload)
(21,Payload) (22,Payload)
(23,Payload) (24,Payload)
(2,Sherpa)
(v:7)
b1-t3
roles:
(4,BaseCamp) (18,Payload)*
(v:1)
LinkCapacity: [∞] (e:8)
LinkCapacity: [8] (e:5)
LinkCapacity: [∞] (e:0)
lander-t4
roles:
(13,Payload) (14,Payload)
(15,Payload)
(v:10)
b4-t6
roles:
(19,Payload)* (20,Payload)*
(21,Payload) (22,Payload)
(23,Payload) (24,Payload)
(2,Sherpa)
(v:6)
b1-t4
roles:
(4,BaseCamp)*
(v:0)
LinkCapacity: [∞] (e:7)
LinkCapacity: [∞] (e:4)
lander-t5
roles:
(13,Payload)* (14,Payload)*
(15,Payload)*
(v:9)
b4-t7
roles:
(21,Payload) (22,Payload)
(23,Payload) (24,Payload)
(2,Sherpa)
(v:5)
LinkCapacity: [∞] (e:3)
LinkCapacity: [8] (e:2)
b4-t8
roles:
(22,Payload)* (23,Payload)*
(24,Payload)*
(v:4)
b6-t10
roles:
(21,Payload)* (2,Sherpa)*
(v:3)

Figure 10: A final mission outline computed by our planning system from which individual plans can be computed from. The asterisk (*) marks roles that have reached their final destination. Link capacities depend on the robotic agents transferring from one location to another; link capacities to transfer to, i.e. remain at, same location are infinite.

generation of temporal constraint networks to satisfy this requirement. Furthermore, we adopt of a flaw-based plan repair strategy similar to (Maio et al. 2015). The planning approach has been validated using a set of example mission specifications.

This paper currently only mentions the use of metrics to analyse and optimize multi-robot plans, but our main motivation of the planning approach aims at using the information about the organizational, i.e. multi-robot system's, state to optimize resource usage and distribution. In general, our future work will be focused on completion of the full planning approach including the integration of the multi-objective optimization and an application of the planning system to the real reconfigurable multi-robot system (cf. Figure 1).

Since planning can only operate on an abstraction of the real world, any resulting plan will leave potential for optimization. However, to augment the multi-robot planning approach we suggest to strengthen the ability for local collaboration, e.g., in terms of annotating a solution plan with potential for local optimization. While the multi-robot plan identifies two or more agents that join at some location and point in time, this information should be used by this subset of agents to join at an even earlier stage, e.g., by sharing knowledge about target positions, leading to an embedded, online and local optimization.

6 Acknowledgments

This work was supported by the German Space Agency (DLR) under grant agreement 50RA1301 and by the Federal Ministry of Education and Research under grant agreement 01IW15001.

References

Ahuja, R. K.; Magnanti, T. L.; and Orlin, J. B. 1993. *Network Flows: Theory, Algorithms, and Applications*. Michigan, US: Prentice Hall.

Baca, J.; Hossain, S.; Dasgupta, P.; Nelson, C. a.; and Dutta, A. 2014. ModRED: Hardware design and reconfiguration planning for a high dexterity modular self-reconfigurable robot for extra-terrestrial exploration. *Robotics and Autonomous Systems* 62(7):1002–1015.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; and Carreras, M. 2015. ROSPlan: Planning in the Robot Operating System. In *Proceedings of International Conference on AI Planning and Scheduling (ICAPS)*, 333–341.

Chien, S. A.; Johnston, M.; Frank, J.; Giuliano, M.; Kavelaars, A.; Lenzen, C.; and Policella, N. 2012. A generalized timeline representation, services, and interface for automating space mission operations. In *Proceedings of the 12th International Conference on Space Operations*, 1–17. Reston, Virigina: American Institute of Aeronautics and Astronautics.

Dechter, R. 2003. Temporal Constraint Networks. In Dechter, R., ed., *Constraint Processing*, The Morgan Kaufmann Series in Artificial Intelligence. San Francisco: Morgan Kaufmann. chapter 12, 333–362.

DeLoach, S. A., and Kolesnikov, V. A. 2006. Using Design Metrics for Predicting System Flexibility. In Baresi, L., and Heckel, R., eds., *Fundamental Approaches to Software Engineering*, volume 3922. Springer Berlin Heidelberg. 184–198.

Dignum, V., ed. 2009. *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI Global.

Eich, M.; Hartanto, R.; Kasperski, S.; Natarajan, S.; and Wollenberg, J. 2014. Towards coordinated multirobot missions for lunar sample collection in an unknown environment. *Journal of Field Robotics* 31(1):35–74.

Evans, J. S. 1991. Strategic flexibility for high technology manoeuvers: A conceptual framework. *Journal of Management Studies* 28:69–89.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2012. Using the context-enhanced additive heuristic for temporal and numeric planning. *Springer Tracts in Advanced Robotics* 76(STAR):49–64.

Gantner, Z.; Westphal, M.; and Wölfl, S. 2008. GQR-A fast reasoner for binary qualitative constraint calculi. In *Proceedings of the AAAI'08 Workshop on Spatial and Temporal Reasoning*, volume 8, 24–29.

Horridge, M., and Bechhofer, S. 2011. The OWL API: A Java API for OWL ontologies. *Semantic Web* 2(1):11–21.

Hübner, J. F.; Sichman, J. S. a.; and Boissier, O. 2002. MOISE+. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems - Part 1*, AAMAS '02, 501. New York, USA: ACM Press.

Kennington, J. L. 1978. A Survey of Linear Cost Multicommodity Network Flows. *Operations Research* 26(2):209–236.

Maio, A. D.; Fratini, S.; Policella, N.; Donati, A.; and Agency, E. S. 2015. Resource driven planning with plasma: the plan space multi-solver application. In *Proceedings of the 13th Symposium on Advanced Space Technologies in Robotics and Automation*.

Meiri, I. 1996. Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence* 87(1-2):343–385.

Murphy, R.; Ausmus, M.; and Bugajska, M. 1999. Marsupial-like Mobile Robot Societies. *Proceedings of the Third Annual Conference on Autonomous Agents* 1–14.

NASA. 2016. Mission Timeline: Surface Operations, retrieved May 3, 2016, from `http://mars.nasa.gov/mer/mission/tl_surface_nav.html`.

Nau, D.; Ghallab, M.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Rahwan, T.; Ramchurn, S. D.; Jennings, N. R.; and Giovannucci, A. 2009. An Anytime Algorithm for Optimal Coalition Structure Generation. *Journal of Artificial Intelligence Research* 34:521–567.

Rahwan, T.; Michalak, T. P.; Elkind, E.; Faliszewski, P.; Sroka, J.; Wooldridge, M.; and Jennings, N. R. 2011. Constrained Coalition Formation. In *Proceedings of Twenty-First AAAI Conference on Artificial Intelligence*, 719–725.

Roehr, T. M.; Cordes, F.; and Kirchner, F. 2014. Reconfigurable integrated multirobot exploration system (rimres): Heterogeneous modular reconfigurable robots for space exploration. *Journal of Field Robotics* 31(1):3–34.

Schulte, C., and Tack, G. 2012. View-based propagator derivation. *Constraints* 18(1):75–107.

Sonsalla, R.; Cordes, F.; Christensen, L.; Planthaber, S.; Albiez, J.; Scholz, I.; and Kirchner, F. 2014. Towards a Heterogeneous Modular Robotic Team in a Logistic Chain for Extraterrestrial Exploration. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Stock, S.; Mansouri, M.; Pecora, F.; and Hertzberg, J. 2015. Hierarchical Hybrid Planning in a Mobile Service Robot. In Hölldobler, S.; Krötzsch, M.; Penaloza, R.; and Rudolph, S., eds., *KI 2015 – Advances in Artificial Intelligence*. Springer International Publishing. 309–315.

Tenorth, M., and Beetz, M. 2013. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research* 32(5):566–590.

Tsarkov, D., and Horrocks, I. 2006. FaCT++ Description Logic Reasoner: System Description. In Furbach, U., and Shankar, N., eds., *Automated reasoning*, volume 4130 of *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg. 292–297.

Weiss, G., ed. 2009. *Multiagent Systems*. MIT Press, 2nd edition.

Wurm, K. M.; Dornhege, C.; Nebel, B.; Burgard, W.; and Stachniss, C. 2013. Coordinating heterogeneous teams of robots using temporal symbolic planning. *Autonomous Robots* 34(4):277–294.

Zhong, C., and DeLoach, S. 2011. Runtime Models for Automatic Reorganization of Multi-Robot systems. In *Proceedings of the 6th Symposium of Software Engineering for Adaptive and Self-Managing Systems*, 20 – 29. Honolulu, HI, USA: ACM Press.