Towards Hybrid Reasoning for Automated Industrial Fleet Management

Masoumeh Mansouri, Henrik Andreasson, Federico Pecora
Center for Applied Autonomous Sensor Systems,
Örebro University, SE-70182 Sweden.
`<name.surname>@oru.se`

Abstract

More and more industrial applications require fleets of autonomous ground vehicles. Today's solutions to the management of these fleets still largely rely on fixed set-ups of the system, manually specified ad-hoc rules. Our aim is to replace current practice with autonomous fleets and fleet management systems that are easily adaptable to new set-ups and environments, can accommodate human-intelligible rules, and guarantee feasible and meaningful behavior of the fleet. We propose to cast the problem of autonomous fleet management to a meta-CSP that integrates task allocation, coordination and motion planning. We discuss design choices of the approach, and how it caters to the need for hybrid reasoning in terms of symbolic, metric, temporal and spatial constraints. We also comment on a preliminary realization of the system.

1 Introduction

More and more industrial applications require fleets of autonomous ground vehicles. These include intralogistics, construction sites, and mining. In all of these industries, it is required to develop fleets and fleet management systems that can respond to changes rapidly. In intralogistics, for example, fleets of autonomous forklifts have to adapt to changing warehouse layout, the addition/removal of forklifts, dynamic priorities, and new storage and shelving areas. Fleet behavior must meet certain operational requirements. Ideally, we would like to specify these requirements at a high-level of abstraction and have the robots autonomously adapt and act appropriately.

However, today's fleet management solutions for autonomous vehicles still largely rely on fixed set-ups of the system (e.g., pre-defined trajectories for robots navigation [Marshall *et al.*, 2008]) and manually specified ad-hoc rules and functions (i.e., specifying ad-hoc traffic rules for autonomous safe navigation). Current approaches present two major drawbacks: first, even small modifications to the environment require to re-think the overall process (e.g., defining new trajectories and manually updating traffic rules, both of which are expensive and lengthy procedures); second, it is difficult to guarantee desired formal properties (e.g., ad-hoc traffic rules are not sufficient to guarantee deadlock-free coordination). The first drawback motivates us to design an autonomous system that is easily adaptable to new set-ups, environments and human-intelligible rules. The second drawback suggests an automated solution to fleet management whose formal properties can be proven, thus guaranteeing feasible and meaningful behavior of the system.

The knowledge representation and reasoning (KR&R) underlying an automated fleet management system should be *hybrid*. The representation should be symbolic, as this facilitates the job of modeling system behaviors. It should also accommodate metric models since various reasoning processes are done at the metric level (e.g., motion planning). In addition to accommodating qualitative and metric knowledge, the representation should encompass various semantics that allows for different types of reasoning (e.g., temporal and spatial reasoning). For example, consider a scenario where vehicles have to accomplish tasks that require moving from their current positions to goal positions in such a way that their maneuvers are collision free. In this system, three different sub-problems have to be solved: task allocation, motion planning and coordination. Solutions to each sub-problem depend on the solutions to other sub-problems. In other words, it is necessary to subject the possible choices made to solve one problem to the choices made in resolving the other problems. For example, a particular decision about an allocation of a robot to a task may result in a motion that is in conflict with that of another vehicle. In this case, coordination may decide to delay the other vehicle to avoid the conflict, which may lead to the violation of a deadline constraint on a delivery of a particular task, that in turn will affect the task allocation decision. In general, to solve an overall hybrid problem, we need to interleave individual reasoners (e.g., motion planning and coordination) to account for inter-dependencies among their decisions. This can be feasible through a hybrid representation that is shared among reasoners for posting the consequence of theirs decisions.

To address the issues above, we work toward building a system for autonomous fleet management that facilitates hybrid reasoning by casting the high-level requirement of a system to a meta-CSP. More specifically, we propose a solution to the management of fleets of autonomous vehicles that (1) incorporates task allocation (2) accounts for individual feasible motions of each vehicle with respect to the vehicle's kinematic model; (3) avoids deadlocks and collisions; (4) is able to accommodate externally imposed temporal and spatial constraints that are either metric or qualitative.

In this paper, the terms robot and vehicle are used interchangeably, and an autonomous vehicle will be referred to as a vehicle for brevity.

2 Representation

Our system is grounded upon the notion of *task*. A task represents what operations a vehicle must carry out, as well as where and when these are to occur. The spatial and temporal dimensions of a task are represented as *trajectory envelopes*. These are collections of spatial and temporal constraints on vehicle trajectories and temporal constraints on when the task is to be achieved. This representation builds upon a purely spatio-temporal representation of trajectories presented in Pecora *et al.* [2012]. Also, the construction of trajectory envelopes is adapted to accommodate spatial constraints at different levels of granularity. The latter helps us to increase scalability for employing less constraints in the reasoning process where appropriate.

2.1 Trajectory Envelopes

A trajectory envelope is composed of a *spatial envelope* and a *temporal envelope*. The former is a set $\mathcal{S} = \{S_1, \ldots, S_n\}$ of constraints on the position and orientation of a vehicle (i.e., a pose) within a given map M of the environment. Each S_i is convex polygon resulting from the convex hull of a set of polyhedral constraints $\{c_i^1, \ldots, c_i^m\}$, where each c_i^j is a set of linear inequalities that bound the vehicle's pose.

To each S_i we associate a set of temporal constraints T_i of the form

$$\ell_i \leq e_i - s_i \leq u_i \tag{1}$$

where s_i (e_i) denotes the time in which the vehicle's pose begins (ceases) to be within the polyhedral constraint S_i. $\ell_i, u_i \in \mathbb{R}$ are fixed lower and upper bounds. (1) defines bounds on when the reference point of the vehicle is within the convex region specified by S_i. The bounds ($\ell_{i,i+1}, u_{i,i+1} \in \mathbb{R}$) on when the reference point is within the spatial overlap between S_i and S_{i+1} is defined as

$$\ell_{i,i+1} \leq e_i - s_{i+1} \leq u_{i,i+1}, \tag{2}$$

The temporal envelope of a vehicle is the collection of temporal constraints $\mathcal{T} = \{T_1, \ldots, T_n\}$.

Definition 1. *A* trajectory envelope *is a pair $\mathcal{E} = (\mathcal{S}, \mathcal{T})$, where*

- $\mathcal{S} = \bigcup_i S_i$ *is the* spatial envelope *of the vehicle, and*
- $\mathcal{T} = \bigcup_i T_i$ *is the* temporal envelope *of the vehicle.*

A trajectory envelope is thus a set of spatial and temporal constraints on a vehicle's trajectory — that is, each pair (S_i, T_i) constitutes a polyhedral constraint (polyhedron for short) that curtails the possible poses of the vehicle and when the vehicle can assume those poses.

Let $\boldsymbol{p} : [0,1] \to \mathbb{R}^2 \times \mathbb{S}^1$ denote a path for a vehicle, in terms of positions and orientations of its reference point, parametrized using its arc length σ. Given a time history along the path $\sigma = \sigma(t)$, we refer to $\boldsymbol{p}(\sigma)$ as a trajectory. $\mathcal{E}$ contains the trajectory $\boldsymbol{p}(\sigma)$ if $\boldsymbol{p}(\sigma(t)) \in S_i$ for all $t \in T_i$. The number of S_i over a given trajectory $\boldsymbol{p}(\sigma)$ is determined by the number of changing points computed by the Ramer-Douglas-Peucker (RDP) [Douglas and Peucker, 1973] algorithm given a parameter ϵ. Higher values of ϵ lead to a lower number of polyhedra $S_i \in \mathcal{S}$. Henceforth, we use $(\cdot)^{(j)}$ to indicate that variable $(\cdot)$ is associated to the j-th vehicle. Given N vehicles, each with trajectory envelope $\mathcal{E}^{(j)}, j \in \{1 \ldots N\}$, the problem of finding $s_i^{(j)}$ and $e_i^{(j)}$ (for all i, j) that satisfy temporal constraints $\{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(N)}\}$ is Simple Temporal Problem (STP) [Dechter *et al.*, 1991] with variables $\boldsymbol{t} = \bigcup_{i,j}\{s_i^{(j)}, e_i^{(j)}\}$. A STP admits many solutions, each defining the amount of time $e_i^{(j)} - s_i^{(j)}$ during which vehicle j's reference point should be within the polyhedral constraint $S_i^{(j)}$. A solution of this STP can be found in $\Theta(|\mathcal{S}|^3)$ with the Floyd-Warshall all-pairs-shortest-paths algorithm [Floyd, 1962], where $\mathcal{S} = \sum_{j=1}^{N} \left|\mathcal{S}^{(j)}\right|$.

So far, we have explained the metric level spatial and temporal constraints subject to which the vehicles are to navigate the common Map M. Both types of constraints are given as sets of inequalities. Temporal inequalities in $\mathcal{T}$ and spatial inequalities in $\mathcal{S}$ form the basic constructs for metric temporal and spatial reasoning. However, in order for humans to impose requirements on vehicle behavior, it is useful to be able to specify qualitative spatial and temporal constraints: for example, the occurrence of a certain polyhedron should be before the occurrence of another one, or one polyhedron should be 'disconnected' from another specific polyhedron. In order to reason about such requirements, we provide a set of qualitative constraints with semantics that can be reasoned about at the metric level.

Temporal CSP. The elements T_i of a trajectory envelope can be seen as variables in a qualitative temporal CSP. Their domain is an interval $[s_i, e_i]$ where s_i represents the start time and e_i represent the end time of the interval. Constraints are basic Allen relations or a convex disjunction of Allen relations [Allen, 1984]. The former are the thirteen possible temporal relations between intervals, namely "before" (b), "meets" (m), "overlaps" (o), "during" (d), "starts" (s), "finishes" (f), their inverses (e.g., b^{-1}), and "equals" ($\equiv$). For example, we can state that T_i is in relation "overlaps" with T_j. These qualitative relations have temporal semantics that can be expressed as inequalities between start/end times. Hence, reasoning about qualitative constraints can be performed in the STP.

Spatial CSP. The elements S_i of a trajectory envelope can be seen as variables in a qualitative spatial CSP. Constraints in this CSP include the constraint DC (disconnected).DC is a directed binary constraint between two polyhedra. Imposing A DC B results in moving the polyhedron A such that it is not no longer connected to polyhedron B and B DC A results in moving B to enforce disconnection. At the metric

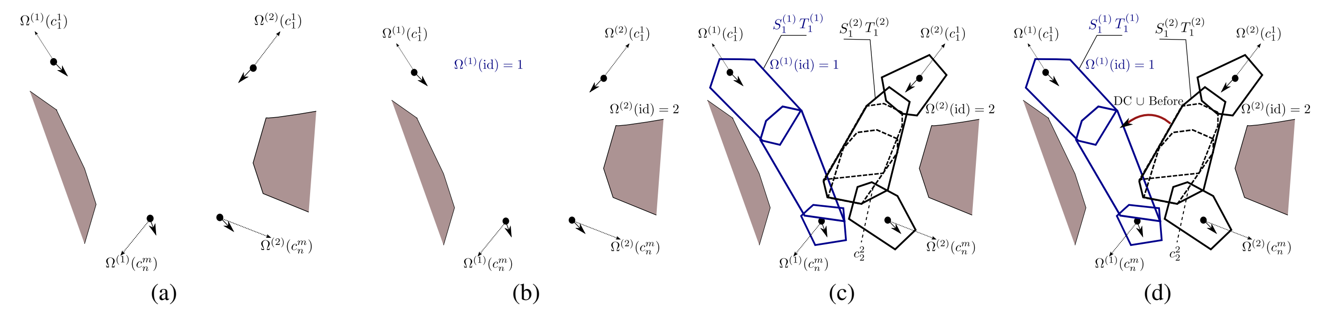

Figure 1: An example of two tasks with their targets and trajectory envelopes.

level, DC translates a polyhedron (i.e., moves every point of a polyhedron constant distance in a specified direction) by a vector. The translation function given by vector v is T_v. We calculate the penetration vector (pv) through SAT (Separating Axis Theorem) [Boyd and Vandenberghe, 2004] and use it as a translation vector to enforce the minimum displacement required to disconnect the two polyhedra. There are DC constraints between trajectory envelopes and all obstacles. Therefore, enforcing a DC constraint may result in an inconsistency in the spatial CSP if the displaced polyhedron intersects an obstacle. In summary, we define a high-level spatial CSP over the metric representation to provide a representational framework which can be extendable to various types of symbolic spatial constraints. Employing different high-level constraints, such as cardinal constraints or constraints in Region Connection Calculus (RCC), is a topic for future work. Performing high-level spatial reasoning (e.g., path-consistency over a network of RCC relations) avoids unnecessary metric operation in case of inconsistency at the higher level.

2.2 Vehicle Target

In addition to trajectory envelopes, other types of knowledge that are used in the fleet management process must be represented.

Definition 2. *A* vehicle target *is a tuple* $\Omega = \big(ID, OP_s, OP_e, c_1^1, c_n^m, C_t, C_s, LOAD\big)$, *where* ID *represents the ID of the robot set to perform* OP_s *and* OP_e *at respectively starting pose* c_1^1 *and goal pose* c_n^m *under a set of user-defined temporal constraints* C_t *and spatial constraints* C_s.

A vehicle can perform various types of operations, namely, loading, unloading, load detect (i.e., the cargo location should be detected using perception) and can have different types of loads. The types of load and operation are inputs to the system. C_t are C_s are metric or qualitative spatial and temporal constraints. Henceforth, we denote with V the set of all vehicle IDs, and with $\Omega(\cdot)$ an element of the target tuple.

2.3 Task Network as Hybrid Representation

The core of our representation is a *task*. A task serves as variable whose values encompass all the requirements to which a vehicle is subject to:

Definition 3. *A* Task *is a pair* $\mathcal{K} = (\Omega, \mathcal{E})$, *where* $\mathcal{E} = (\mathcal{S}, \mathcal{T})$ *is the* trajectory envelope *of the vehicle, and* Ω *is the* target *of the vehicle.*

Reasoning employs a common constraint network, called *task constraint network*, which is used as a common search space for solving the hybrid problem. Its variables are tasks and its constraints are spatial and temporal constraints.

Tasks and constraints can be used to represent the requirements subject to which vehicles should operate. For instance, Figure 1(d) the trajectory envelopes of two vehicles, as well as the requirement that the two vehicles are either spatially or temporally separated in the narrow passage. The latter requirement is expressed as a disjunction of a spatial and a temporal constraint. In the following section, we describe how these requirements are obtained incrementally via reasoning in the joint search space of different sub-problems.

3 Constraint Based Reasoning for Industrial Vehicles

A task network constitutes a common representation for reasoning in several modules, namely, *meta-CSP search*, *trajectory extractor* and *vehicle executive*.

Meta-CSP search decides allocations of vehicle IDs to tasks (i.e., which vehicle performs a task), and generates trajectory envelopes that are guaranteed to be collision-free, deadlock-free, adherent to spatial and temporal constraints, and traversable by the vehicles. The meta-CSP search does so by exploring the joint search space posed by the task allocation, motion planing and coordination problems.

A *trajectory extractor* extracts trajectories for a given task $\mathcal{K}^{(j)}$ from its trajectory envelopes $\mathcal{E}^{(j)}$.

A *vehicle executive* realizes the interface between vehicle controllers and the trajectory envelope representation by updating the trajectory envelopes with constraints representing the current progress of each vehicle. This propagates any mismatch between prescribed and executed trajectories on all vehicles in the fleet. A *controller* on board each vehicle synthesizes control actions according to a Model Predictive Control (MPC) scheme [Qin and Badgwell, 2003].

3.1 Meta-CSP Search

The tasks and constraints in a task network describe the possible behaviors of vehicles. For the purposes of the industrial settings that are the focus of our work, we require task networks to be *feasible*:

Definition 4. *A task network is* feasible *iff*

- *all tasks are assigned to vehicles;*
- *all tasks include feasible motions, i.e., for each task $\mathcal{K}^{(j)}$, there is a set of motions $\mathcal{E}^{(j)}$ leading vehicle j from its current position to a goal position;*
- *all tasks are deadlock- and collisions-free;*
- *the temporal CSP and spatial CSP are consistent.*

Obtaining a task network that is feasible is a problem at the higher level of abstraction. The decisions leading to feasible task networks consist of incrementally adding variables and constraints to the task network. This combinatorial problem is solved by a CSP-like search, in a meta-level CSP (henceforth called meta-CSP).

A meta-CSP is a CSP whose constraints, called meta-constraints, are first three high-level requirements stated above. In particular, meta-constraints in our application restrict decisions related to task allocation, motion planning, and coordination. The latter, for instance, restricts trajectory envelopes so that they are collision free. Meta-CSP search interleaves reasoning about the three meta-constraints to guarantee that all requirements are upheld simultaneously. The meta-CSP is defined at a higher level of abstraction of the temporal and spatial CSPs, and it uses these CSPs to propagate the high-level decisions made for satisfying the meta-constraints. Note that spatial and temporal consistency is decided without backtracking through temporal and spatial propagation. Hence, the fourth requirement is not a meta-constraint.

Similarly to a CSP, a meta-CSP is characterized by its constraints, variables and variable domains (values). A meta-variable is a subset of the task network that constitutes a *flaw* (or infeasibility) with respect to some high-level requirement — e.g., a pair of polyhedra in two trajectory envelopes that are temporally and spatially overlapped constitute a collision. Meta-values are the different possibilities to resolve a flaw; for example, to resolve a flaw in the task network pertaining to the collision of two vehicles, temporal constraints can be posted into the task constraint network forcing one of the vehicles to slow down to prevent the collisions; alternatively, a spatial constraint can be imposed to force one of the vehicles to take a new path.

Search in meta-CSP consists in finding an assignment of resolvers (meta-values) to flaws (meta-variables) given a set of meta-constraints and the task network. When all flaws are resolved, the task network represents a feasible set of tasks. We use a CSP-style backtracking search to find meta-values to be added to the task network. Meta-variables to branch on are chosen based on variable ordering heuristics, and alternative meta-values are chosen according to value-ordering heuristics. Next, we explain each meta-constraint and its contribution to solving the overall hybrid problem.

3.2 Task Allocation Meta-Constraint

The task allocation meta-constraint imposes complete assignments of vehicles to tasks. An example is shown in Figure 1(b). Its meta-variables are tasks that do not have an assigned robot ID, i.e., $\{\mathcal{K}^{(j)} \mid \Omega^{(j)}(\mathrm{id}) = \emptyset\}$. Meta-values are assignments of robot IDs to targets, i.e., $\Omega^{(j)}(\mathrm{id}) \in V$.

In the present stage of development, we use a naïve way of allocating robots to tasks. In general, casting task allocation as a meta-constraint has several advantages. We can use any off-the-shelf task allocator to solve the problem and this can be done without changing the representation. The task allocation meta-constraint is a part of meta-CSP search, not merely a pre-processing step. Therefore, various alternatives can be tried based on the choices made by other meta-constraints. For instance, if a particular assignment to a task creates a deadlock with respect to robot motions, another feasible assignment can be tried. More importantly, the shared representation can provide a good heuristic for task allocation, since the task network includes spatial and temporal knowledge, thus, an estimation of the completion time of previous tasks and the current and future states of robots. All of the points above become more important when re-allocation has to be done efficiently on-line when contingencies occur. This is even more so if the problem has to be solved for up to hundreds of vehicles.

3.3 Motion Planning Meta-Constraint

The motion planning meta-constraint decides trajectory envelopes that adhere to given kinematic constraints (see, e.g., Figure 1(c)). Its meta-variables are tasks that have been assigned a vehicle ID but not a trajectory envelope, that is, $\{\mathcal{K}^{(j)} \mid \Omega^{(j)}(\mathrm{id}) \neq \emptyset \wedge \mathcal{E}^{(j)} = \emptyset\}$. Meta-values are trajectory envelopes $\mathcal{E}^{(j)} = (\mathcal{S}^{(j)}, \mathcal{T}^{(j)})$.

The spatial envelope $\mathcal{S}^{(j)}$ of a meta-value contains one or more kinematically feasible paths that lead the vehicle from its current pose to the goal pose $\Omega^{(j)}(c_n^m)$. $\mathcal{S}^{(j)}$ contains the positions and orientations along a nominal path computed by a motion planner, as well as adjacent positions and orientations obtained by "sweeping" the footprint of the vehicle within given displacement and turning parameters. In this work, we used a lattice-based motion planner [Cirillo *et al.*, 2014b]. The resulting spatial envelope gives the vehicle controller the freedom to spatially deviate from the reference trajectory while remaining within the spatial constraints. The temporal envelope $\mathcal{T}^{(j)}$ is calculated for each vehicle, and contains the constraints (1) and (2). The motion planner provides the state information and boundary conditions on initial and final velocities, steering velocities, speed and acceleration. Based on the maximum allowed velocity and distance between two states, a minimum transition time Δt_{min} is computed. This is used to compute the lower bounds in the temporal envelope $\mathcal{T}$. The maximum transition time Δt_{max} determines the upper bounds. We impose a minimum speed $v_{\min} = \epsilon > 0$: this allows vehicles to move arbitrarily slow, a condition which is needed to ensure schedulability of joint motions [Pecora *et al.*, 2012].

3.4 Spatio-temporal Coordination Meta-Constraint

A coordination meta-constraint refines the envelopes $\{\mathcal{E}^{(1)}, \ldots, \mathcal{E}^{(N)}\}$ of all vehicles so as to eliminate trajectories which lead to collisions or deadlocks (see example in Figure 1(d)). These may occur because vehicles share the same floor space, hence rendering it necessary to impose that trajectory envelopes do not overlap in both time and space. Meta-variables of this meta-constraint are pairs of polyhedra represented by quadruple $(S_k^{(i)}, S_m^{(j)}, T_k^{(i)}, T_m^{(j)})$, of vehicle i and j respectively, that overlap both spatially and temporally, i.e.,

$$S_k^{(i)} \cap S_m^{(j)} \neq \emptyset \wedge \tag{3}$$

$$T_k^{(i)} \cap T_m^{(j)} \neq \emptyset. \tag{4}$$

The meta-value of a meta-variable is a set of four constraints:

$$\{S_k^{(i)}\{DC\}S_m^{(j)}, S_m^{(j)}\{DC\}S_k^{(i)}, \tag{5}$$

$$T_k^{(i)}\{\text{before}\}T_m^{(j)}, T_m^{(j)}\{\text{before}\}T_k^{(i)}\} \tag{6}$$

Adding one of the above constraints to the task network resolves the flaw by removing either conditions (3) or condition (4). For example if the meta-CSP search selects a temporal constraint among (6) for resolving a flaw, and this turns out to be inconsistent with other temporal constraints in the task network, then backtracking may lead it to select a spatial resolver among (5).

When temporal constraints are added to the task network, their metric semantics are added to the underlying STP, that is, the inequalities $\left\{s_k^{(i)} \geq e_m^{(j)}, e_k^{(i)} \leq s_m^{(j)}\right\}$. Temporal constraint propagation updates the start and end points of the temporal variables $(T_k^{(i)}, T_m^{(j)})$, which in turn delays one of the vehicles. This approach is the Earliest Start Time Approach precedence-constraint posting algorithm [Pecora *et al.*, 2012] for vehicle coordination.

If a spatial constraint $S_k^{(i)}\{DC\}S_m^{(j)}$ is chosen as a resolver for a flaw, $S_k^{(i)}$ is translated along the penetration vector, thus eliminating spatial intersection (see Section 2.1). If two trajectories cross each other, we exclude meta-values (5), restricting flaw resolution to temporal coordination. We say that two trajectories cross each other if their linear interpolations cross each other.

This meta-constraint employs a variable-ordering that gives preference to pairs of polyhedra which have bigger intersecting areas. The value-ordering heuristic prefers sequencing choices that maximize temporal flexibility [Cesta *et al.*, 2002].

In summary, meta-CSP search branches over choices (meta-values) for solving flaws (meta-variables) pertaining to all three meta-constraints. It terminates when there is no more flaw in the task network to be resolved. In the next Section, we will explain how the trajectory extractor selects a trajectory from the trajectory envelopes in the task network. The trajectory extractor operates after meta-CSP search terminates, hence the extracted trajectory is guaranteed to be feasible. The whole system is on-line and meta-CSP search is called at a frequency of 1Hz to check for new flaws.

4 Trajectory Extractor

The task network obtained from the meta-CSP search contains trajectory envelopes that are consistent with temporal and spatial constraints and are deadlock and collision-free. Now, trajectories that are traversable by the robots have to be extracted from the envelopes. The trajectory extractor generates highly accurate, smooth and drivable trajectories which respect the dynamic and kinematic constraints of the vehicles [Andreasson *et al.*, 2015]. This is done through an optimization process in which the objective function is to minimize the total distance traveled and the total amount of turning applied on the steering wheel. The objective function is subject to a set of constraints, including those that constitute the trajectory envelopes. This guarantees that trajectories conform to the kinematic constraints of the vehicle as well as to constraints on the vehicle controls. In addition, there are also constraints that force the initial and end states generated by the trajectory extractor be exactly equal to the actual state of the vehicle and given goal poses. The trajectory extractor is an on-line process and the key to its efficiency is to start the search from a cost-optimal and obstacle free path. We employ for this purpose the path that was generated as a starting point for computing spatial envelopes (see Section 3.3).

5 Discussion

We perform several test runs in a simulated industrial production site (using Gazebo) using simulated forklifts. Each test run opens up interesting issues, which we discuss in the remainder of this section.

Preliminary experiment. The first scenario is a show-case for the use of spatio-temporal coordination where new spatial constraints are imposed as meta-values to the common constraint network and result in avoiding future collisions of vehicles. Figure 2 shows snapshots of the system. The task allocation meta-constraint identifies two upcoming targets as meta-variables and assigns vehicles to the targets as the meta-values (a). The motion planning meta-constraint identifies two tasks whose trajectory envelopes are empty as meta-variables. The meta-values are trajectory envelopes shown in (b). The spatio-temporal coordination meta-constraint identifies a pair of polyhedra in which conditions (3) and (4) hold (see Figure 2(b) for the temporally and spatially overlapped polyhedra). The meta-value is a spatial constraint DC whose imposition between these two conflicting polyhedra moves one of them to the right, thus removing condition (3), which resolves the collision flaw (c). The trajectory extractor extracts a trajectory for each vehicle which adheres to the constraints that are present in the task network (see red lines in Figure 2(c)). The vehicle controllers follow the extracted trajectories within the limits of the envelopes, and the execution interface updates the status of the task network as vehicles proceed (d).

The second scenario is a show-case for the spatio-temporal coordination of two vehicles (see Figure 3(a) and 3(b)). Figure 3(a) shows a situation where both robots are assigned to the targets such that their trajectory envelopes cross each other. The spatio-temporal meta-constraint identities pairs of overlapping polyhedra in space and time. Conditions (3) and (4) constitute a flaw. The resolvers are the two alternative 'before' constraints whose imposition between conflicting polyhedra would remove the condition (4). As a consequence of the chosen 'before' constraint, robot2 is delayed

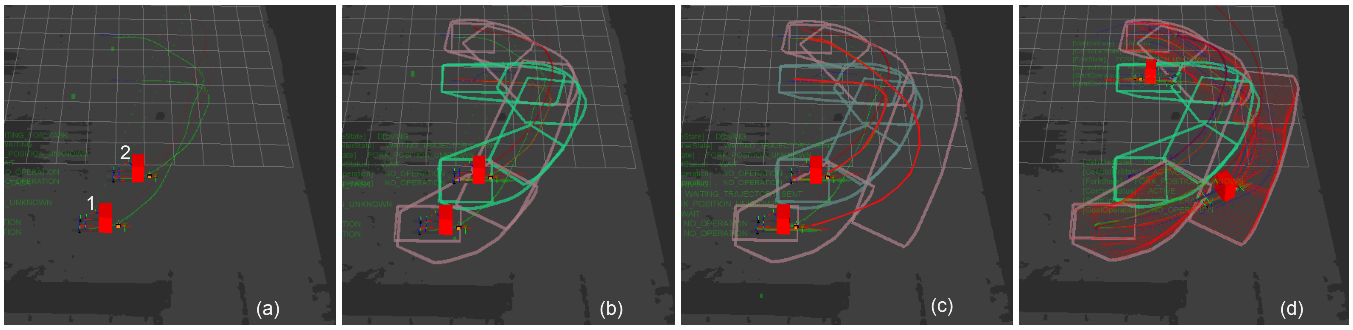

Figure 2: An example of spatio-temporal coordination, where a DC constraint is added as a resolver to solve the spatio-temporal flaw.

until robot1 passes the conflicting areas. Figure 3(b) shows the moment in which robot2 starts moving toward its goal pose after robot1 is no longer in the conflicting areas.

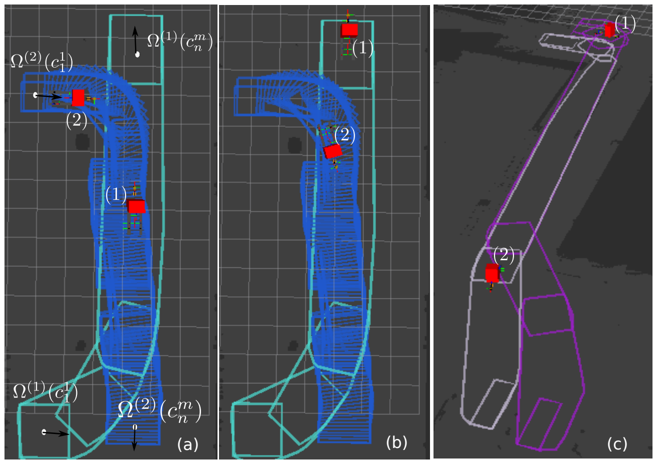

Figure 3: Examples of spatio-temporal coordination.

Scalability and Flexible Representation. Figure 3(c) depicts a situation where two robots have to drive through a long corridor in order to reach their goal poses. The trajectory envelopes of robot1 and robot2 are shown in the figure. As explained in Section 2.1, each S_i is a convex hull of a set of smaller polyhedra $\{c_i^1, \ldots, c_i^m\}$. The size of this set is defined by the RDP algorithm which depends on the changing point of the initial trajectory, and can be different for each S_i. The aggregation of smaller polyhedra to a big polyhedron entails trajectory envelopes with fewer temporal variables $\mathcal{T}$. This facilitates temporal constraint propagation, which in turn positively affects scalability.

Similarly to the second scenario, one robot has to wait for another one as a consequence of the temporal constraints posted by the spatio-temporal coordination. The important issue here is that robot1 waits for robot2 in order to leave the intersecting polyhedra. However, the intersecting polyhedra cover almost the entire corridor. This is not a desirable situation since many industrial applications are interested in decreasing waiting time. This problem can be easily solved by increasing the size of trajectory envelopes, i.e., decreasing the parameter ϵ used in the RDP algorithm. Increasing the size of trajectory envelopes affects scalability. The temporal CSP continuously updates the temporal bounds over the trajectory envelopes, and increasing the number of its variables slows down temporal propagation.

The trade-off between trajectory envelope size and scalability suggests employing a flexible representation, i.e., maintaining trajectory envelopes at different levels of granularity, and selecting an appropriate level based on the types of resolvers. For example, we can employ convex hulls of fewer small polyhedra c_i^j to create S_i when the coordination meta-constraint chooses DC constraints to resolve flaws; conversely, we can increase granularity when it chooses to perform temporal flaw resolution.

Traffic Rules. Figure 3(c) points to another important issue to be discussed. A narrow wall in the corridor was added in the physical setup of the factory to facilitate lane management for human drivers. When deploying autonomous vehicles and automated fleet management solutions, one could avoid the realization of such physical infrastructure by exploiting spatial constraints to model lane management requirements and other symbolic traffic rules. For example, the long corridor can be divided into two polyhedra with opposite orientation. Each polyhedron represents a lane with a desired direction (i.e., a lane polyhedron). A meta-constraint can be added that is responsible for lane management. In this meta-constraint, the meta-variables are trajectory envelopes which spatially intersect a lane polyhedron, and the difference in their orientations is higher than a threshold. In other words, a flaw is a pair of polyhedra, one being a lane polyhedron and the other a polyhedron in the trajectory envelope of a vehicle, whose orientation bounds are in opposite directions. A meta-value for such a flaw would be a spatial constraint enforcing the trajectory envelope be in the lane polyhedron next to it (i.e., the correct lane). Notice that the trajectory extractor then extracts the drivable trajectory that adheres to the updated trajectory envelopes; all other meta-constraints, as well as the search algorithm, also remain identical. Hence no further change to the algorithm or representation is required to account for lane specifications.

5.1 Related work on Hybrid Reasoning

Hybrid reasoning in AI started with systems based on Krypton [Brachman *et al.*, 1985]. Subsequently Description Logic (DL) and its extensions earned the reputation of being hybrid. These KR formalisms initially included various forms of qualitative reasoning. Nowadays, hybrid reasoning is mainly about combining qualitative and metric reasoning. For example, combining task planing (a form of qualitative reasoning) with motion planning (a form of metric reasoning) is considered as a hybrid reasoning problem, and is an active research topic. In robotics, approaches to hybrid reasoning are mainly focused on how to include a general high-level reasoner (e.g., a classical task planner) with specialized reasoners (e.g., motion planner). Kaelbling and Lozano-Pérez [2011] and Garrett *et al.* [2014] give an overview on different approaches of combining task and motion planning. Our approach falls in the category of hybrid reasoning in a robotic domain, specifically because we employ various types of reasoning at metric and qualitative levels, and because our shared constraint-based representation contains various semantics of the system, which enables us to interleave high-level as well as specialized reasoners.

In our work, the meta-CSP includes a meta-constraint responsible for the coordination of multiple vehicles in a centralized fashion. Distributed motion coordination using polynomial-time distributed algorithms (only at the metric level) is used to guarantee safe navigation [Bekris *et al.*, 2012]. As opposed to entirely metric reasoning, Saribatur *et al.* [2014] employ ASP as a high level representation and reasoning schema augmented with external calls to specialized metric reasoners for the coordination of teams of heterogeneous robots. The latter work is interesting for us, as it uses ASP as a convenient way to encode high-level specifications. Including an ASP program in a meta-constraint, thus using an ASP solver within a meta-CSP search, is one of our ongoing directions of work.

Cirillo *et al.* [2014a] uses the same technique for motion planing and trajectory extractor as ours. We augment the coordination techniques used in this paper with the spatial resolver, and also include task allocation as a part of integrated reasoning algorithm.

6 Conclusion and Ongoing Work

We have presented a fleet management system which adheres to several key requirements. Our approach is grounded on a common constraint-based representation which is shared among various reasoners. This common representation includes both metric and qualitative knowledge enabling reasoning at both levels. The constraints capture heterogeneous requirements of a wide category of industrial domains. The core algorithm is meta-CSP search which interleaves reasoning to achieve a feasible solution with respect to all requirements. The focus of our ongoing work is to evaluate whether the tight integration of different modules through the meta-CSP approach outperforms existing fleet management systems [Cirillo *et al.*, 2014a]. Future work will also include developing specific applications of the approach in underground and surface mining applications, intralogistics and construction site management.

Acknowledgments. This work is supported by the Swedish Knowledge Foundation (KKS) project "Semantic Robots".

References

J.F. Allen. Towards a general theory of action and time. *Artif. Intell.*, 23(2):123–154, 1984.

H. Andreasson, J. Saarinen, M. Cirillo, T. Stoyanov, and A. Lilienthal. Fast, continious state path smoothing to improve navigation accuracy. In *IEEE Conference on Robotics and Automation (ICRA)*, 2015.

K. E. Bekris, D. K. Grady, M. Moll, and L. E. Kavraki. Safe distributed motion coordination for second-order systems with different planning cycles. *Int. Journal of Robotics Research (IJRR)*, 31(2), 2012.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

Ronald J. Brachman, Victoria Pigman, Gilbert Hector, and J. Levesque. An essential hybrid reasoning system: knowledge and symbol level accounts of krypton. In *In Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 532–539. Morgan Kaufmann, 1985.

A. Cesta, A. Oddi, and S. F. Smith. A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 8(1):109–136, January 2002.

Marcello Cirillo, Federico Pecora, Henrik Andreasson, Tansel Uras, and Sven Koenig. Integrated motion planning and coordination for industrial vehicles. In *Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2014.

Marcello Cirillo, Tansel Uras, and Sven Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, pages 112–122, 1973.

Robert W. Floyd. Algorithm 97: Shortest path. *Communication of the ACM*, 5:345–348, June 1962.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Ffrob: An efficient heuristic for task and motion planning. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical planning in the now. In *IEEE Conference on Robotics and Automation (ICRA)*, 2011.

Joshua Marshall, Timothy Barfoot, and Johan Larsson. Autonomous underground tramming for center-articulated vehicles. *Journal of Field Robotics*, 25(6-7):400–421, 2008.

F. Pecora, M. Cirillo, and D. Dimitrov. On mission-dependent coordination of multiple vehicles under spatial and temporal constraints. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

S. Qin and T. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

Z. Saribatur, E. Erdem, and V. Patoglu. Cognitive factories with multiple teams of heterogeneous robots: Hybrid reasoning for optimal feasible global plans. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.