# A Planner for Ambient Assisted Living: From high-level reasoning to low-level robot execution and back*

**M. Di Rocco**[1] and **S. Sathyakeerthy**[1] and **J. Grosinger**[1] and **F. Pecora**[1] and **A. Saffiotti**[1]

**M. Bonaccorsi**[2] and **F. Cavallo**[2] and **R. Limosani**[2] and **A. Manzi**[2] and **G. Teti**[3] and **P. Dario**[2] † ‡ §

**Abstract**

Robot ecologies are a growing paradigm in which one or several robotic systems are integrated into a smart environment. Robotic ecologies hold great promises for elderly assistance. Planning the activities of these systems, however, is not trivial, and requires consideration of issues like temporal and information dependencies among different parts of the ecology, exogenous actions, and multiple, dynamic goals. We describe a planner able to cope with the above challenges. We show in particular how this planner has been incorporated in closed-loop into a full robotic system that performs daily tasks in support of elderly people. The full robot ecology is deployed in a test apartment inside a real residential building, and it is currently undergoing an extensive user evaluation.

## 1 Introduction

Robotic ecologies (Saffiotti et al. 2008) are networked systems composed of one or several robots that cooperate with static sensors and actuators deployed in the environment. The combination of the advanced motion and manipulation capabilities afforded by robots, and the pervasive sensing and actuation capabilities afforded by the smart environment, often results in systems which are more robust, flexible, and modular than a traditional monolithic robot. Because of this, robot ecologies hold great promises for applications in domestic, everyday environments like elderly assistance. In fact the increase of life expectancy and reduction of births has induced a rapid increase of elderly population and consequently a number of societal challenges, such as provision of social and medical services with sustainable costs and extension of the working life of senior citizens.

Different works in literature address the integration of robots and smart environments to provide physical and cognitive support in the field of ambient assisted living (AAL). In (Schroeter et al. 2013), a companion robot is used to cope with mild cognitive impairments. (Huijnen et al. 2011) describes two projects devoted to developing platforms for assisting people in daily tasks such as reminding or entertainment. In (Lowet and Frank 2012) an inexpensive architecture has been developed to provide a support similar to the previous approaches. (Cavallo et al. 2013) presents a multi function robot for physical and cognitive assistance at home. (Bedaf et al. 2013) and (Cousins 2011) describe an infrastructures equipped with robotic manipulators to provide physical support to the users. Relevant results have also been obtained with outdoor or office-like platforms (Ferri et al. 2011; Kanda et al. 2009; Mizoguchi et al. 1999). In most of these works, however, it is assumed that the service provided by the system consists of a single task; where this is not the case, the form of reasoning employed to cope with multiple tasks is simplified by not considering situations where there might be multiple and concurrent goals. Furthermore, the above approaches do not consider multiple robots.

In this paper we present a planner for multi-robot, multi-task ecologies developed in the context of the EU project Robot-Era.[1] To the best of our knowledge, this is the first attempt to integrate an autonomous multi-robot system in a complex, smart environment in the service of elderly people. This application domain introduces several important new challenges and requirements to the planning system: (a) since we do not focus on one specific task, the planner must cope with a variety of different, possibly concurrent goals that can be posted any time; (b) because of the dynamic nature of the environment and of human interaction, the planner is committed to perform high-level reasoning that has to be further mapped into low-level (inter-)actions that fulfill the on-going requirements; (d) the presence of a multi-robot system demands a form of coordination between the units that can possibly perform heterogeneous actions while receiving support from the smart environment; and (d) the intrinsic concurrency in a robot ecology rises challenges related to the usage of common resources, time synchronization, and causal and information dependencies.

To satisfy the above requirements, we control our robot ecology using a configuration planner (M. Di Rocco et al. 2013) that is able to produce fine-grained plans for robotic

*This work was funded by the EC Seventh Framework Programme (FP7/2007-2013) grant agreement no. 288899 Robot-Era.

†1Center for Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden. maurizio.di-rocco@oru.se

‡2The BioRobotics Institute, Scuola Superiore Sant'Anna, Pisa, Italy. f.cavallo@sssup.it

§3Robotech srl, Italy. g.teti@robotechsrl.com



[1] www.robot-era.eu

systems. The planner can model the causal, temporal, resource and information dependencies between the sensing, computation, and actuation components in one or multiple robots. As we shall see, the planner has features which are geared toward the integration with low-level, physical robot execution in the real world. In this paper we show how the configuration planner has been integrated in the full RobotEra robot ecology, starting from high-level reasoning and refining this down to low-level interaction between the devices in the ecology.

## 2 Related work

Classical planning approaches make a number of assumptions that abstract away from the details of physical execution in real, dynamic environments (Ghallab et al. 2004). In particular, aspects of time, information, and resources are often ignored. By contrast, our work is grounded on temporal planning techniques, and enriches this paradigm with the ability to reason about resource, causal and information dependencies among the entities involved in the plan. All these features are needed when orchestrating networked robotic systems: the information flow plays a crucial role to coordinate the various units, while resource constraints must be carefully taken into account since devices may share and exploit limited resources. The temporal aspect has a pivotal role, and the tasks involved may be subject to deadlines. In addition, the concurrency of multiple goals emphasizes the role of synchronization between interacting functionalities.

Today's planners exhibit some of the above features. For instance, a considerable amount of work has been done to integrate metric time into planning (Knight et al. 2001; Do and Kambhampati 2003; Gerevini et al. 2006; Doherty et al. 2009; Barreiro et al. 2012; Eyerich et al. 2009). Including time has allowed some planning systems to perform *continuous planning*, i.e., continuously synthesize plans as new goals and contingencies become known. Execution monitoring techniques have been developed which leverage the explicit temporal representation of these plans (Finzi et al. 2004; McGann et al. 2008b; 2008a), thus effectively providing a few examples of planning for real robots. Although these constitute important steps towards obtaining planners that are appropriate for robots, they do not address the need to represent and reason about other aspects of physical domains, like space, information and resources.

Some work has addressed the issue of including resources into the planning problem. Some of these approaches (Köckemann et al. 2012; Fratini et al. 2008; Ghallab and Laruelle 1994) would also be well suited for use in closed loop with actuation and perception, as they maintain a certain level of least-commitment with respect to the timing of plan execution. Nevertheless, they are neither defined nor evaluated as closed-loop planning systems. (Lemai and Ingrand 2004) propose an extension of the IxTeT planner (Ghallab and Laruelle 1994) for closed loop execution monitoring with resources; however, their technique is only applied to single robot navigation tasks.

## 3 High-level reasoning

Task planners typically reason about abstract causality: what actions should be performed in order to bring the system in a state where the goal is satisfied? To better close the gap with low-level robot execution, our configuration planner also reasons about finer-grained aspects that are relevant to successful execution, like time, information, and resources. Moreover, our planner operates in closed-loop: it continuously incorporates new sensor observations and goals during execution, and it adapts the current plan to take these into account — or, if adaptation is not possible, generates an entirely new plan.

Our approach is grounded on the notion of *state variable* (SV), which models elements of the domain whose state in time is represented by a symbol. State variables, whose domains are discrete sets, represent features of the environment we want to model as well as functionalities provided by the ecology. The particular instantiation of a functionality, called *activity*, can either produce information (*information output*) or effects on the world, i.e., act on state variables representing the environment. We employ a (flexible) temporal representation that allows us to restrict the occurrence in time of a particular predicate over a SV.

To be executed, a functionality may need sources of information (*information input*) and consume resources. Our framework takes into account *reusable resources*, i.e., resources that are fully available when not used by any functionality. The information flow as well as the manipulation of the world leads to the connection among different functionalities during plan generation: these interactions and dependencies are modeled temporally through Allen's Interval Algebra (Allen 1984). These 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., $\mathrm{b}^{-1}$), and "equals" ($\equiv$). For example, the relation $f_2 \ \{\mathrm{m}\} \ f_1$ represents that $f_2$ ends as soon as $f_1$ starts.

**Definition 1** *A* constraint network *is a pair* $(\mathcal{A}, \mathcal{C})$*, where* $\mathcal{A}$ *is a set of* activities *and* $\mathcal{C}$ *is a set of* temporal constraints *among activities in* $\mathcal{A}$*.*

Plans are constraint networks in which activities are linked by temporal relations generated by causal, resource and information constraints (Di Rocco et al. 2013b). We call these *configuration plans*, as they describe the desired behavior of the overall system in time, including both actuation and information generating components:

**Definition 2** *A* feasible configuration plan *is a constraint network that is:*

- temporally consistent*, i.e., there exists at least one allocation of fixed bounds to intervals such that all temporal constraints are satisfied;*
- resource feasible*, i.e., activities do not over-consume resources over time;*
- symbolically feasible*, i.e., state variables are not prescribed to assume different values at the same time;*

- information feasible*, i.e., for each* information input *required in the plan there exists a supporting activity in the network producing a consistent* information output.

A *goal* can be seen as a constraint network which is not feasible, i.e., whose inconsistencies must be solved by a reasoning system. Such a constraint network implicitly represents a set of desired states — all those that are consistent with the constraints.

Our framework employs the so called Meta-CSP approach: the problem of refining the goal constraint network into a feasible one is cast as a high-level Constraint Satisfaction Problem (CSP) (Tsang 1993) whose variables represent the infeasibilities above. The values of these high-level variables constitute possible ways to solve the infeasibilities, and are computed by a set of specialized solvers. This approach naturally adheres to the necessity of managing several features (temporal, causal, resource conflict) that are of interest to the physical execution of robotic tasks.

### 3.1 Constraint-Based Search

The planning process used in our approach is incremental in nature, and yields a refined constraint network, which itself represents a feasible configuration plan which achieves the given goal. The resulting constraint network represents one or more temporal evolutions of the state variables that guarantee the achievement of the goal under nominal conditions. Feasible and goal-achieving configuration plans are obtained in our approach by means of four interacting solvers:

**Temporal solver.** The temporal consistency of the constraint network is checked through temporal constraint propagation by means of a Simple Temporal Problem (STP) (Dechter et al. 1991) solver. The solver propagates temporal constraints to refine the bounds of the activities in the network, and returns failure if and only if temporally consistent bounds cannot be found.

**Resource scheduler.** This solver enforces that resources are never over-consumed. The maximum capacities of resources restrict which activities can occur concurrently, and this solver posts temporal constraints to the constraint network enforcing that over-consuming peaks of activities are avoided (Cesta et al. 2002).

**State variable scheduler.** State variable scheduling ensures that activities do not prescribe conflicting states in overlapping intervals. Similarly to the resource scheduler, this solver posts temporal constraints which impose a temporal separation between conflicting activities.

**Information dependency reasoner.** Operators model the information dependencies between functionalities[2]. This solver instantiates into the constraint network relevant operators (in the form of activities and temporal constraints) so as to enforce the information dependencies.

**Causal reasoner.** Operators in the domain also model causal dependencies between states. This solver instantiates into the constraint network relevant operators (in the form of activities and temporal constraints) so as to enforce the causal dependencies of the configuration plan.

[2]In our approach, the domain is such that information dependencies constitute an acyclic propositional Horn theory.

Technical details about these reasoners can be found in (Di Rocco et al. 2013b).

As noted, resource over-consumption and multiple concurrent states are averted by imposing temporal constraints which sequence potentially concurrent activities; trivially, there are alternative sequencing decisions that can be made to resolve such a conflict, e.g., by enforcing the "before" constraint $a\,\{b\}\,b$ or $a\,\{b^{-1}\}\,b$. Also operator selection is subject to alternative choices, as more than one operator may provide the necessary information output and/or support the necessary causal dependency; e.g., the presence of light in the environment may be obtained as a result of invoking the light controller or by opening the blinds. Only temporal feasibility enforcement is not subject to multiple choices, which makes the problem tractable. In our approach, all requirements which may entail alternative courses of action are seen as *decision variables* in a high-level CSP. Given a decision variable $d$, its possible values constitute a finite domain $\delta^d = \{(A_r^d, C_r^d)_1, \ldots, (A_r^d, C_r^d)_n\}$, whose values are alternative constraint networks, called *resolving constraint networks*. The individual solvers are used to determine resolving constraint networks $(A_r^d, C_r^d)_i$, which are iteratively added to the goal constraint network $(A_g, C_g)$.

Function `Backtrack` $(A_g, C_g)$ : success or failure

```
1  d ← Choose((A_g, C_g), h_var)
2  if d ≠ ∅ then
3      δ^d = {(A_r^d, C_r^d)_1, ..., (A_r^d, C_r^d)_n}
4      while δ^d ≠ ∅ do
5          (A_r^d, C_r^d)_i ← Choose(d, h_val)
6          if (A_g ∪ A_r^d, C_g ∪ C_r^d) is temporally consistent then
7              return Backtrack(A_g ∪ A_r^d, C_g ∪ C_r^d)
8          δ^d ← δ^d \ {(A_r^d, C_r^d)_i}
9      return failure
10 return success
```

In order to search for resolving constraint networks, we employ a systematic search (see Algorithm `Backtrack`), which occurs through standard CSP-style backtracking. The decision variables are chosen according to a variable ordering heuristic $h_{\text{var}}$ (line 1); the alternative resolving constraint networks are chosen according to a value ordering heuristic $h_{\text{val}}$ (line 5). The former decides which (sub-)goals to attempt to satisfy first, e.g., to support a functionality by applying another operator, or to resolve a scheduling conflict. The latter decides which value to attempt first, e.g., whether to prefer one operator over another. Note that adding resolving constraint networks may entail the presence of new decision variables to be considered.

All five solvers employed in the algorithm above are implemented on top of the Meta-CSP framework (Pecora et al. 2012). The framework facilitates the integration of solvers through Meta-CSP techniques, and in particular provides implementations of inter-solver interfaces and overall search modules (such as Algorithm `Backtrack`).

### 3.2 Plan Execution and Dynamic Plan Update

The ability to support on-line sensing is directly enabled by the constraint-based representation: sensing is reduced to dynamically updating the constraint network with new activities and constraints representing the sensed state of the environment; the same mechanism also supports prediction (i.e., "sensing in the future") and other on-line plan modifications, such as temporal delays and dynamically posted goal constraint networks. Activities and constraints that represent sensed information can have different levels of abstraction: low-level observations like "stove-on", filtered state estimates like "user-at-kitchen", or high-level interpretations like "user-is-cooking". Section 4.1 will show the sensing abstractions used in the Robot-Era system.

Our approach to continuous planning is based on the alternation of planning and plan execution monitoring. The former consists of the planning procedure shown above. The latter consists of two processes: *sensing* and *plan update*. The sensing process adds to the constraint network activities and temporal constraints representing the current view of the environment as provided by sensors. The plan update process maintains and updates temporal constraints which bound on-going activities (sensed states or functionalities in execution) with the current time. This is done in $O(n^2)$ through incremental temporal constraint propagation (Dechter et al. 1991), where $n$ is the number of activities in the constraint network. Also, this process imposes constraints that verify the existence of preconditions and trigger the manifestation of effects contained in the plan. Specifically, the presence of a precondition is verified by attempting to unify the activity representing the precondition with a sensed activity. If the unification is not possible, the precondition is delayed by inserting a temporal constraint, and is re-evaluated at the next iteration. The process enforces the occurrence of activities representing effects by posting temporal constraints which fix their start time to the current time. The effect of the constraints posted by these processes is that functionalities start when possible, are delayed until the preconditions hold, and their effects are imposed when necessary. This step also requires polynomial computation.

In our current implementation, all solvers monitor the network for new decision variables. Thus "re-planning" occurs by temporal propagation, resource or state variable scheduling, or operator application, depending on the situation.

Note that this allows to keep the computational impact of replanning at a minimum (e.g., operator application need not occur if scheduling is sufficient, scheduling need not occur if temporal propagation is sufficient). This mechanism is what enables dynamically posted goals, as in other temporal constraint-based continuous planners (McGann et al. 2008b; Barreiro et al. 2012), but here we also deal with resources, sensor data and information constraints.

All the components so far described post activities and/or constraints into the temporal network and their relations can be compared to the ones existing between components of a classical control system. The dynamic goal posting corresponds to the desired state for the system to control; in order to achieve this state (which changes over time) several solvers manipulate the temporal network, which can be seen as a way of formulating high-level control signals. Once decisions are taken, control signals are injected into the state if they did not lead to temporal inconsistencies (validation performed by the temporal solver). Finally, the state of the world is continuously fed back to the system through the observer. A schematic representation of this comparison is depicted in Fig. 1. We show an example of this behavior in the next Section.

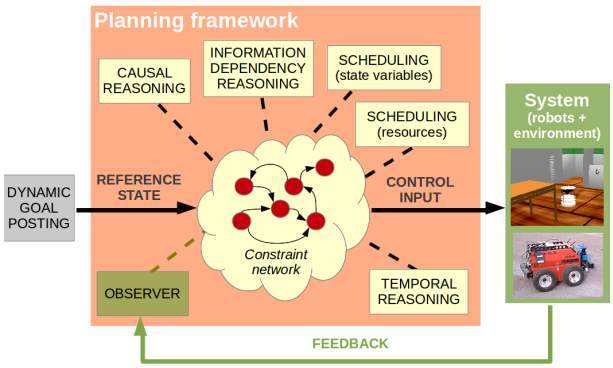


Figure 1: High level reasoners (causal reasoner, information dependency reasoner, schedulers) modify the constraint network to achieve the dynamically changing desired state (dynamic goal posting). Their decisions are temporally validated (temporal reasoning) and sent to the system for execution. The current state of the system is continuously maintained in the constraint network (observer).

## 4 The Robot-ERA System

The above planner has been implemented in a specific module, called Configuration Planning Module (CPM) and included in the full Robot-ERA system. This is a networked robot system whose structure can be conceptually divided into sensing and actuation modules. The former provide, at the moment, a minimal form of context awareness that may automatically trigger goals in addition to the ones explicitly requested by the user; the latter allows manipulation of the world to fulfill the requirements of a configuration plan.

All the components in the system communicate by means of the PEIS Middleware (Saffiotti and Broxvall 2005), which implements a distributed blackboard model: each module publishes and reads from a decentralized tuple-space (see Fig. 2). The PEIS Middleware allows devices to join or leave the network dynamically. This mechanism constitutes the interface between CPM and the software modules associated to the activities in the plan.

### 4.1 Sensing Modules

The sensing abstraction provided to the high-level reasoning is grounded in the information produced by sensors deployed in the environment. Currently, we employ wireless sensor network nodes (Motes) equipped with heterogeneous sensors: passive infrared (PIR), pressure, humidity, light, temperature and proximity transducers. Each of them is installed in meaningful positions (e.g., a pressure sensor under a chair) to provide pieces of information that processed

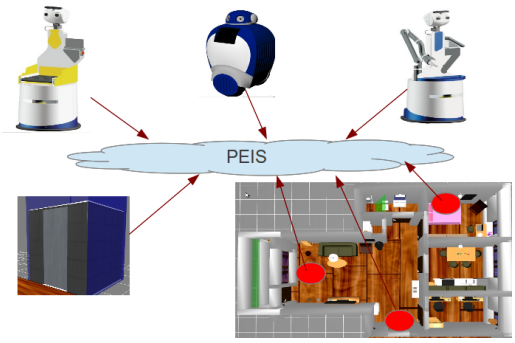


Figure 2: Blackboard model: robots, actuators and sensors communicate by means of the PEIS Middleware.

together provide an abstraction for one or multiple environmental variables.

In this paper we focus on a particular subsystem which estimates the position of the user. We have used two complementary methods: the first relies on the RSSI signal produced by a subset of Motes; the user, wearing a small module, is localized by means of trilateration techniques (Cavallo et al. 2011). The output of the algorithm provides the (metric) location of the user within the apartment. The second system is based on pressure and PIR sensors (see Fig. 3); the (temporal) alternation of signals can provide a topological location but also can infer simple behaviors (e.g., having the bed-light OFF *during* bed-pressure ON entails that the user is sleeping). At the moment this reasoning process is still minimalistic and is currently under development, as is the fusion between the metric and discrete sources of information.

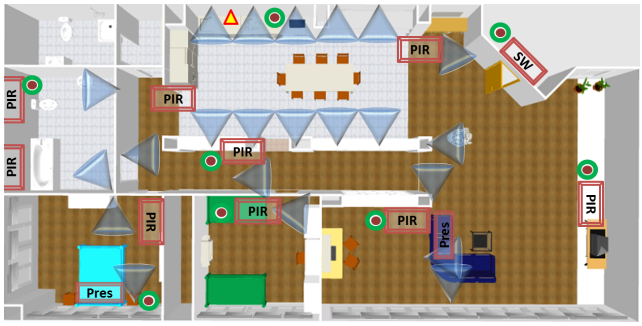


Figure 3: Map of the Domo Casa at the Biorobotics institute. The environment is equipped with antennas to detect the position of the human and with heterogeneous transducers to detect interesting features of the environment. Several PIR, pressure (PRES), temperature (circles) and gas (triangles) are installed in the house

Information from sensors is continuously fed back to the planner in the form of new activities with fixed start and evolving end times. Through the process of unification explained earlier, this knowledge is used by the CPM to derive plans that are contextualized to the current state as it is sensed by the sensors.

Figure 4: The three platforms used in the experimentation: the outdoor robot *Oro* (left), the condominium robot *Coro* (center) and the domestic robot *Doro* (right),

### 4.2 Actuation

The actuation capabilities are mostly performed by robots, with some exceptions in the case of simple forms of actuation like turning on lights or moving between floors by means of an automated elevator.

In our project we employ three robots: two Metralabs SCITOS G5 platforms, and one RoboTech Dustcart. Each robot is equipped with ROS, and its functionalities are modeled in the domain of the planner. The execution of an activity is often bounded to an instantiation of a functionality that communicates by PEIS with the CPM. As we show in Sec. 5, the planner's symbolic reasoning processes (temporal inference, planning, resource and state variable scheduling, and information dependency reasoning) result in lower-level actuation commands dispatched in metric time.

For example, necessary synchronizations between robots and resource conflicts are mapped into timely dispatching of actuation commands: suppose that a robot is required to move from the first to the sixth floor to access the laundry room. While from a high level perspective the task is simply to travel the edge of a topological graph, this action is further refined involving the elevator and several software modules. Those that are responsible for actuating the robot are therefore activated and synchronized with the ones moving the elevator and its door (Fig. 5).

## 5 Experiments

In this section we describe several runs of the system for the purpose of highlighting the interesting features of our approach. A quantitative evaluation of the planner alone is outside the scope of this paper and can be found in (Di Rocco et al. 2013b).

The CPM was employed in two different test-beds: the DomoCasa Living Lab of the Biorobotics Institute in Italy (see Fig. 6) and the Ängen Smart Home in Sweden (see Fig. 7). The two facilities are prototypical smart homes built for the same purpose, developing robot ecologies for elder care, and similar robots were deployed in both. However, the two facilities are structurally different and populated by different sensors and actuators. The CPM is currently used to

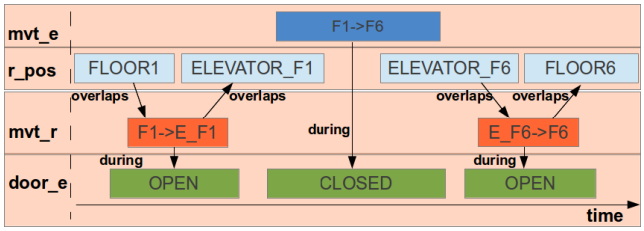


Figure 5: Moving between floors. The motion of the robot (*mvt_r*) from the floor to the elevator (*F1→E_F1*) and vice-versa (*E_F6→F6*) can happen only when the elevator door is open (*during* constraint). Similary the motion of the elevator (*mvt_e*), supporting the motion of the robot from the 1st to the 6th floor (*F1→F6*) can only happen when the doors are closed. The execution of functionalities can modify the state variables representing environmental state, e.g., the motion of the robot modifies the state of *r_pos*. The state representing the position of the elevator and the modules acting on the door are not shown for simplicity.

evaluate eleven case-studies in a long-term field evaluation within the EU project Robot-ERA.[3].

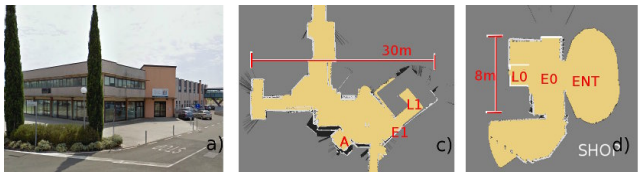


Figure 6: DomoCasa Living Lab of the Biorobotics Institute (Scuola Superiore Sant'Anna, Italy). The Biorobotics Institute (left); ROS map of the first floor (middle); ROS map of the ground floor (right). Both maps shot the symbolic locations used in the plan.

## 5.1 Synchronization: goods exchange

This experiment describes an example of synchronization between two robots. The scenario evolves as follows:

> *Sven has ordered some goods at the nearby supermarket; he lives at the first floor of a building where a condominium robot,* Coro*, and an outdoor robot,* Oro*, provide services for the inhabitants.* Oro *is instructed to go the supermarket to collect goods prepared by the clerk. When coming back, it will pass the goods to* Coro *at the entrance of the building.* Coro *will then deliver the goods to Sven; it will interact with the elevator in order to move across floors inside the building.*

Although contingencies are not taken into account, the system has to properly coordinate two robots that have to exchange the package in a given location. One of the robots employs the elevator to go from the user's apartment to the exchange location and back. The experiment was conducted at the Biorobotics lab using the ground (Fig. 6-c) and the first floor (Fig. 6-d). The plan generated is depicted in Fig. 8 while video of the experiment is available at `http://youtu.be/jY74RtufYIo`.

[3]www.robot-era.eu

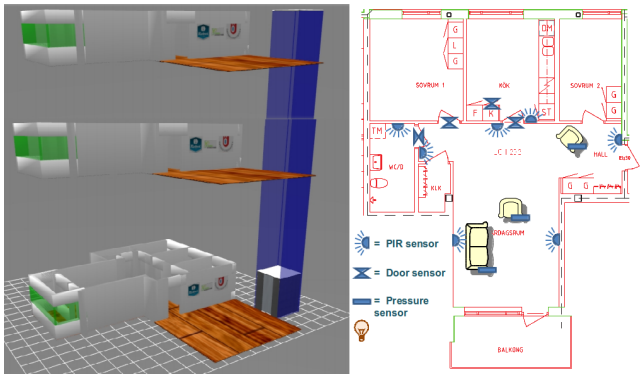


Figure 7: Second experimental setup. Left: a CAD representation of the environment. The scenario runs on three different floors: food is picked up on ground-floor, the user's apartment is on the first floor, and the shared residential laundry room is on the top floor. Right: map of the user's apartment and its sensors.

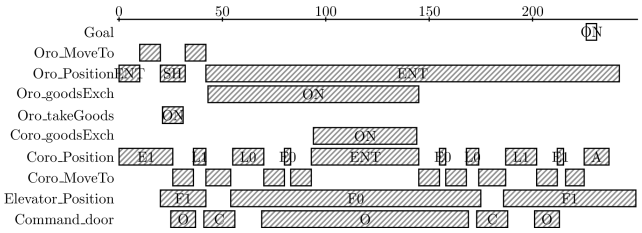


Figure 8: *Oro* moves from the entrance (ENT) to the shop (SH) to fetch goods (Oro_takeGoods) and then comes back. In the meanwhile *Coro* moves from its location (E1) to the elevator at the first floor (L1); the elevator brings *Coro* to the ground floor (L0). Then, it exits the elevator (E0) and finally moves to ENT where the exchange occurs (Coro_goodsExch and Oro_goodsExch). After the completion of the exchange, *Coro* goes upstairs to Sven's apartment (A) to deliver the package (Goal). The motion from one location to another is handled by the MoveTo functionality that can either exploit the Kinect or Laser modules. Note how travelling to and from the elevator implies the opening of the elevator door, and that moving the elevator requires the door to be closed.

## 5.2 Plan adaptation

In this experiment we highlight the flexibility of the reasoner to cope with multiple requests:

> *Sven sends the laundry by* Coro *to the laundry-room at the 6th floor; the robot after executing this task is expected to deliver it back in 1 hour; while delivering the laundry back to Sven, Gunilla's food arrives at the entrance of the building...*

This calls into play the same flexibility necessary to deal with contingencies, as described in (Di Rocco et al. 2013a). In this scenario, we do not want the food to become cold; this could not be avoided if the second task were scheduled

after the accomplishment of the laundry task; fortunately, the robot is capable of accommodating two objects on its tray (namely, the robot's tray is modeled as a resource with capacity two); the planner leverages this resource to insert into the plan for delivering the laundry back to Sven also the task of obtaining the food for Gunilla and bringing it back to her before delivering the laundry to Sven (see Fig.9). This example emphasizes the ability of the planner to re-arrange activities in order to maintain the feasibility of the temporal network.

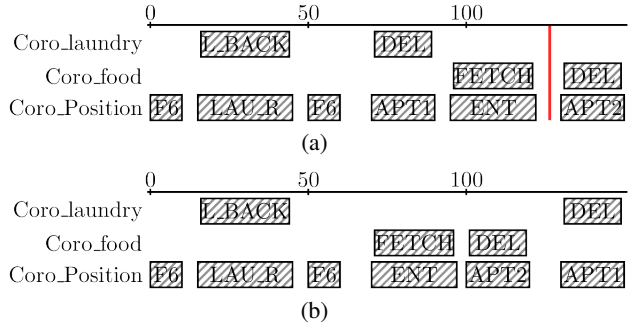


Figure 9: a) the initial plan drives the robot to fetch the laundry (L_BACK) at the laundry-room (LAU_R); afterwards the robot goes to the first apartment (APT1) to deliver the laundry (L_DEL) and then to entrance of the building (ENT) to pick up the food (FETCH) to eventually deliver the food (F_DEL). This plan results inconsistent due to the deadline imposed on the second task b) the alternative plan that achieves both goals: the robot, after visiting the laundry-room, first brings the food to the second user and then delivers the laundry to the first one

## 6 Conclusions

We have described how our constraint-based approach to configuration planning caters for real-world requirements. We have focused on the capability of our system to translate symbolic, high-level requirements (temporal, causal, resource, information requirements) into plans that are executable by a robotic framework. Several experiments were carried out in two different facilities where robots and smart environments are deployed. These preliminary results will be further validated by elderly users in the context of the Robot-ERA project. Eleven different scenarios like the ones shown in this paper will be evaluated through active participation followed by questionnaires.

The inclusion of our planner in a real, deployed and evaluated robotic ecology has allowed us to identify the need for the key technological issues (a)-(d) presented in the Introduction. More issues are also needed, and they will be addressed in our future work. One such issue is the need for manipulation, which will be required in several of our use cases (e.g., picking objects, setting tables). For these, we plan to include a spatial representation into our planning domain. The particular approach described in (Mansouri and Pecora 2013) seems particularly suited as it allows to describe desires spatial layouts in qualitative terms while resulting in metric spatial plans that are directly executable by a robot.

Another open issue is pointed out by the example in Sec. 5 about multiple requests. While that example shows the flexibility of our planner, it also exposes situations that currently are not handled by it. Namely, if the robot had the capacity to carry only one object (laundry or food), and the second goal was launched after the robot picks up the laundry, our planner would not find a solution. This is because the planner cannot achieve goals stated in terms of resrouce usage: in this case, it would have to decide to empty the tray in order to carry the food as a sub-goal of the goal of food delivery. Future work will explore more sophisticated forms of causal planning that cater to resource usage goals.

## References


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

Barreiro, J.; Boyce, M.; Frank, J.; Iatauro, M.; Kichkaylo, T.; Morris, P.; Smith, T.; and Do, M. 2012. EUROPA: A platform for AI planning. In *Proc of ICAPS-ICKEPS*.

Bedaf, S.; Gelderblom, G.; de Witte, L.; Syrdal, D.; Lehnmann, H.; and Dautenhahn, F. A. K. 2013. Selecting services for a service robot. In *ICORR13*.

Cavallo, F.; Aquilano, M.; Bonaccorsi, M.; Mannari, I.; Carrozza, M.; and Dario, P. 2011. Multidisciplinary approach for developing a new robotic system for domiciliary assistance to elderly people. In *Annual Int Conf of the IEEE Eng. in Medicine and Biology Society, EMBC*, 5327–5330.

Cavallo, F.; Aquilano, M.; Bonaccorsi, M.; Limosani, R.; Manzi, A.; Carrozza, M. C.; and Dario, P. 2013. On the design, development and experimentation of the ASTRO assistive robot integrated in smart environments. In *IEEE Int Conf on Robotics and Automation (ICRA)*, 4310–4315.

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

Cousins, S. 2011. ROS on the PR2. In *Springer Lecture Notes in Computer Science - LNCS 7040*, 324–329.

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

Di Rocco, M.; Pecora, F.; and Saffiotti, A. 2013a. Closed loop configuration planning with time and resources. In *Proc of the ICAPS 2013 Workshop on Planning and Robotics*.

Di Rocco, M.; Pecora, F.; and Saffiotti, A. 2013b. When robots are late: Configuration planning for multiple robots with dynamic goals. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Do, M. B., and Kambhampati, S. 2003. Sapa: A multi-objective metric temporal planner. *J. Artif. Intell. Res. (JAIR)* 20:155–194.

Doherty, P.; Kvarnström, J.; and Heintz, F. 2009. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Autonomous Agents and Multi-Agent Systems* 19(3):332–377.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2009. Using the context-enhanced additive heuristic for temporal and nu-

meric planning. In *Proc. of the 19th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Ferri, G.; Manzi, A.; Salvini, P.; Mazzolai, B.; Laschi, C.; and Dario, P. 2011. DustCart, an autonomous robot for door-to-door garbage collection: From DustBot project to the experimentation in the small town of Peccioli. In *IEEE Int Conf on Robotics and Automation (ICRA)*, 655–660.

Finzi, A.; Ingrand, F.; and Muscettola, N. 2004. Model-based executive control through reactive planning for autonomous rovers. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Fratini, S.; Pecora, F.; and Cesta, A. 2008. Unifying planning and scheduling as timelines in a component-based perspective. *Archives of Control Sciences* 18(2):231–271.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An approach to temporal planning and scheduling in domains with predictable exogenous events. *J. Artif. Int. Res.* 25(1):187–231.

Ghallab, M., and Laruelle, H. 1994. Representation and control in IxTeT, a temporal planner. In *AIPS*, 61–67.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Huijnen, C.; Badii, A.; van den Heuvel, H.; Caleb-Solly, P.; and Thiemert, D. 2011. Maybe it becomes a buddy, but do not call it a robot - seamless cooperation between companion robotics and smart homes. In *Springer Lecture Notes in Computer Science - LNCS 7040*, 324–329.

Kanda, T.; Glas, D.; Shiomi, M.; and Hagita, N. 2009. Abstracting people's trajectories for social robots to proactively approach customers. In *IEEE T. on Robotics*, 1382–1396.

Knight, S.; Rabideau, G.; Chien, S.; Engelhardt, B.; and Sherwood, R. 2001. Casper: Space exploration through continuous planning. *Intelligent Systems* 16(5):70–75.

Köckemann, U.; Pecora, F.; and Karlsson, L. 2012. Towards planning with very expressive languages via problem decomposition into multiple csps. In *Proc. of the ICAPS Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS)*.

Lemai, S., and Ingrand, F. 2004. Interleaving temporal planning and execution in robotics domains. In *Proc of the 19th national conference on Artifical intelligence*, 617–622.

Lowet, D., and Frank, H. 2012. Florence - a multipurpose robotic platform to support elderly at home. In *Workshop on Ambient Intelligence Infrastructures (WAmIi)*.

M. Di Rocco, M.; Pecora, F.; Kumar, P.; and Saffiotti, A. 2013. Configuration planning with multiple dynamic goals. In *AAAI Spring Symposium on Designing Intelligent Robots*.

Mansouri, M., and Pecora, F. 2013. TA representation for spatial reasoning in robotic planning. In *Proc. of the IROS Workshop on AI-based Robotics*.

McGann, C.; Py, F.; Rajan, K.; Ryan, J. P.; and Henthorn, R. 2008a. Adaptive Control for Autonomous Underwater Vehicles. In *Proc. of the AAAI Conf on Artificial Intelligence*.

McGann, C.; Py, F.; Rajan, K.; Thomas, H.; Henthorn, R.; and McEwen, R. 2008b. A Deliberative Architecture for AUV Control. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*.

Mizoguchi, F.; Hiraisci, H.; and Nishiyama, H. 1999. Human-robot collaboration in the smart office environment. In *Advanced Intelligent Mechatronics*.

Pecora, F.; Di Rocco, M.; Köckemann, U.; Mansouri, M.; and Ullberg, J. 2012. The meta-csp framework api. Source and binaries available at `http://metacsp.org`.

Saffiotti, A., and Broxvall, M. 2005. PEIS ecologies: ambient intelligence meets autonomous robotics. In *Proc of the joint conference on Smart Objects and Ambient Intelligence*, sOc-EUSAI '05, 277–281.

Saffiotti, A.; Broxvall, M.; Gritti, M.; LeBlanc, K.; Lundh, R.; Rashid, J.; Seo, B.; and Cho, Y. 2008. The PEIS-ecology project: vision and results. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2329–2335.

Schroeter, C.; Volkhardt, S. M. M.; Einhorn, E.; Huijnen, C.; van den Heuvel, H.; van Berlo, A.; Bley, A.; and Gross, H.-M. 2013. Realization and user evaluation of a companion robot for people with mild cognitive impairments. In *IEEE Int. Conf. on Robotics and Automation, ICRA*, 1145–1151.

Tsang, E. 1993. *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego.