Overview of aSyMov: Integrating Motion, Manipulation and Task Planning

Stéphane Cambon, Fabien Gravot, Rachid Alami

Overview of aSyMov: Integrating Motion, Manipulation and Task Planning

Stéphane Cambon, Fabien Gravot and Rachid Alami
{scambon,fgravot,rachid}@laas.fr
LAAS-CNRS, 7, Avenue du Colonel Roche,
31077 Toulouse CEDEX 04, France.

Abstract

In this paper we propose a new and integrated approach to solve planning problems with both symbolic and geometric aspects. Typical concerned domains are intricate manipulation and task planning problems involving multiple robots in three dimensional worlds. The approach involves a task planner that guides a probabilistic roadmap method used to capture the topology of the free space in various contexts. At each step of our hybrid planner aSyMov both symbolic and geometric data are taken into account. Specific predicates are used to link both aspects. Preliminary results obtained by a prototype implementation are shortly presented and discussed.

1 Introduction

One particularly challenging class of problems that we would like to address, involves action planning with strong interaction with 3D manipulations and motions by several mobile-robots and "movable" objects in a constrained environment.

Task planning has often used examples borrowed from robotics like, for instance, Pick&Place scenarii. However, the effective use of practical task planners in robotics has always been limited to domains where it was possible to establish a clear hierarchy between a high-level task planner and a lower level where geometric problems are dealt with. This is clearly insufficient if one wants to tackle realistic robotics problems.

Efforts towards integration of task planning and motion planning [Lozano-Prez *et al.*, 1987] have faced various difficulties due to the lack of an operational link between them. Indeed, the topology of the free space may change when a object is moved (i.e. it may block a passage) or when a robot grasps or releases an object. Indeed, its shape changes and again the topology of the free space changes.

We propose in this paper a new approach to integrate task planning and motion and manipulation planning. This approach is named hybrid because both aspects are treated simultaneously. The task planner guides the probabilistic roadmap methods used to capture the topology of the free space in various contexts. At each step of the hybrid planning process both symbolic and geometric data are taken into account.

The underlying motion planning is based on *Probabilistic Roadmap Methods* (PRMs) [Kavraki and Latombe, 1994; Siméon *et al.*, 2000] that have been adapted to deal with manipulation issues [Alami *et al.*, 1994; Koga and Latombe, 1994; Ahuactzin *et al.*, 1998].

Our contribution is twofold. The first issue concerns the definition of a set of types and predicates that allow to establish a powerful link between states and operators used by a symbolic task planner and a motion and manipulation planning library.

The second issue involves the control of the search process. The hybrid planner develops a control strategy in which there is a compromise between "learning more" given roadmaps (associated to a state) i.e. exploring more deeply various ways to perform a given action in a given state, or select another action. This process is performed "inside" an integrated planner and not in a subsequent lower level geometrical planning step.

We will first present our geometric tools (section 2) and the associated symbolic representation (section 3). Then, we will define the hybrid state composed of symbolic and geometric informations (section 4) . Section 5 will present the aSyMov algorithm. We will finish this article with two illustrated examples.

2 Geometric attributes and tools

In this section we first recall some basic motion planning definitions and then explain some extensions for manipulation and multi-robot used by aSyMov.

2.1 Some definitions

Environment: it is a 3D-world composed of static component and dynamic one: Robots and Objects.
Robots: A robot is defined by a set of solid bodies linked by joints that define their degrees of freedom (dof). An **Object** is a special robot which does not control its dofs.
Configuration: It is an instantiation of the value of the dof of a given system. The PRMs search is performed in the configuration space.
Roadmap: In order to capture the topology of the free space, the motion planner computes an accessibility graph called roadmap. The nodes represent configurations without collision and the edges are valid robot motions.
Connected components: A roadmap can be composed of several connected components. In each one, there is a valid path connecting any two nodes.
PRM: PRMs have proved to be efficient for highly dimensional motion planning problems [Kavraki and Latombe,

1994; Siméon *et al.*, 2000]. They try to connect an initial and final nodes by random sampling of nodes. When a new node is created, they try to connect it to all the connected components of the roadmap. If the initial and final nodes are in the same component, then a solution is found.

2.2 "Pure" Manipulation Planning

A manipulation problem involves two types of motions [Alami *et al.*, 1994], the *transfer* of an object taken by a robot and the *transit* motion of a robot between two grasping positions. A solution to a manipulation problem is a sequence of transit and transfer motions.

Manipulation for one robot and one object

Recent results [Sahbani *et al.*, 2002] have shown the importance of another search space *Grasp* ∩ *Placement* where a robot is grasping an object in a stable position [1]. This work allows to choose intermediate positions and to solve efficiently complex "one robot one movable object" manipulation tasks.

Manipulation for multi-robots

We will use here our extension of the method [Gravot *et al.*, 2002]. The main points of this approach is the notion of robot composition and the use of several specialized roadmaps. The robot composition enables to build new robots (or objects) from other robots (or objects). For instance a table can be composed of a board and a leg for assembly problems, and a robot carrying an object is composed of a robot and an object. With this definition, a transfer motion is a valid motion for the composed robot and we build a roadmap for transfer motions. Consequently, for multi-robots manipulation planning problems we define several specialized roadmaps for each type of motion. Naturally there are links between these roadmaps. Such links correspond to the robot composition. For example, a node in *grasp* ∩ *placement* roadmap can be divided into a node for the object roadmap and a node for the robot alone (*transit*), or can lead to a node for the *transfer* roadmap. This method will have a strong influence on the definition of the predicates of the task planner (§3).

Limitations

For one robot and one object, only the transit motion can be difficult because the robot roadmap does not take into account the object. A specific algorithm must be used to validate or locally change a trajectory to avoid collision with the object. But with multiple robots all the roadmaps built have to be checked to take into account the other movable objects or robots. This is the main difficulty raised by manipulation for multiple robots. We will see in §5.4 how we take this point into account in the search process.

The second difficulty is obviously the size of the search space. We need a method to choose which roadmap to increase and how to do this.

We claim that a symbolic level can guide the search process (§5.3). Indeed, a task planner can propose different actions (and the associated motions) needed to reach a given goal situation. For example, even if there are several robots in the environment, a task planner may find a plan in which only one robot is used to transport an object to its goal position. In such a case, the geometric part of the search will be limited to the robot and object *grasp* ∩ *placement* roadmaps, and no expensive search will be performed in the other robots' roadmaps.

Another advantage in integrating symbolic and geometric models is the possibility to enrich the manipulation context with all the aspects that can be expressed by action planning. The next section deals with the proposed symbolic representation.

[1] $Grasp$ is the set of configurations where a robot grasps or can grasp a given object. $Placement$ is the set of configurations where a movable object can be put.

3 A symbolic representation linked to the geometry

The symbolic representation has two goals. first it is to provide a symbolic representation of the different robots, objects and roadmaps to guide the search for a solution. Second, to increase the expressiveness of the system with purely symbolic actions and predicates.

For our representation we use the PDDL 2.1 language [Fox and Long, 2001] in order to be able to use a state of the art task planner. In the following, we call a *basic* problem, a planning problem that has only geometrical constraints. We define here the minimal set of specific predicates for such a type of problem. These predicates require a specific treatment.

We use three principal types of symbolic objects: robot, obj (a particular type of robot) and position. The following fixed predicates describe links between those symbolic object.

- **compose ?r1 ?r2 ?r3**: the composition of two robots ?r1 and ?r2 is possible and forms a third robot ?r3.
- **belongs-to ?p ?r ?road-type**: a position ?p belongs to the roadmap of type ?road-type of a robot ?r. The ?road-type variable can be instantiated by *transit*, *transfer* or *grasp* ∩ *placement*.
- **is-specific-pos ?p ?pos-type**: to declare the specificity of a position ?p. Different ?pos-type of specificity are possible: In fact the *goal* and *init* position are specific because they can represent an unique configuration of the robot's degree of freedom. Moreover, this predicate is used to specify areas in which a robot must be to apply a pure symbolic action.
- **path ?p1 ?p2**: it is possible to find a valid path between two positions ?p1 and ?p2 which belong to the same roadmap.
- **connection ?p1 ?p2**: it is possible to find a connection between two positions ?p1 and ?p2 which does not belong to the same roadmap.

Only one fluent predicate is necessary.

- **on ?r ?p**: the robot ?r is situated at the symbolic position ?p.

With this set of predicates, we can develop actions which add or remove "*on*" predicates. Generally, we work with three main actions: *goto* (motion), *pick* (composition), *place* (decomposition). Note that these actions are not hard-coded but are only the result of the application of the predicates described above.

Other symbolic predicates can be added and associated to actions that have no effect at the geometric level. For instance, a predicate *"have-magnetic-key"* can be used as a precondition to an *open-door* action.

4 State and geometric positions

We discuss in this section how the state description that we propose takes into account symbolic as well as geometric representations.

4.1 State

The state can be divided into three parts. The first is purely symbolic; it consists of all the symbolic predicates that are not described in §3. The second is the interface between the task planner and the motion planner composed of the *(on ?r ?p)* predicates. It allows to describe where the robots are at a symbolic level: the robot positions correspond to sets of positions that satisfy a number of attributes. The third part is the corresponding geometric position of the robots and objects in the 3D world.

The use of symbolic positions in the state description provides flexibility to the planner and allows a progressive refinement process.

4.2 Symbolic position

A symbolic position represents the attributes that a robot (or object) position has to satisfy. For example:

```
(belongs-to P1 R1 transit)
(link P1 P2)
(composed R1-carry-O1 R1 O1)
(belongs-to P2 R1-carry-O1 grasp)
(link P2 P1)
```

`P1` describes a position of robot `R1` that allows it to take the object `O1` (and consequently to switch from a transit roadmap to transfer roadmap). This position is used any time `R1` wants to take or drop the object `O1`. So if `P1` appears several time in the plan it can correspond to different configurations in the environment.

It is also possible, for a same motion type, to define several symbolic positions. For instance, in order to help the geometric planner, one can define a symbolic position for each room and use the `(path ?p1 ?p2)` to give a connectivity graph. In our case, we use this capacity only when symbolic constraints are also used to change the position, for example, the number of robots in each room, or the fact that the robot must have a key to open a door...

4.3 Geometric position

Whenever the symbolic position changes, a new geometric position is created. This is done through an accessibility list and a list of constraints. The accessibility list represents all the nodes that can be reached from the previous state. The constraints list represents all the non-collision constraints that must be satisfied for the position. To have a valid geometric position, we need to find at least one node in the accessibility list that satisfies all the constraints.

This description allows to change the configuration when new constraints are added, or when the accessibility list is changed due to the roadmap learning process. For instance, if the planning process has chosen to place an object at a valid node N1, but this node forbids further motion, we can switch to an another valid node N2 in the accessibility list that lets robots move. This change is done in the geometric level and has no influence in the state symbolic level.

5 aSyMov: an hybrid planner algorithm

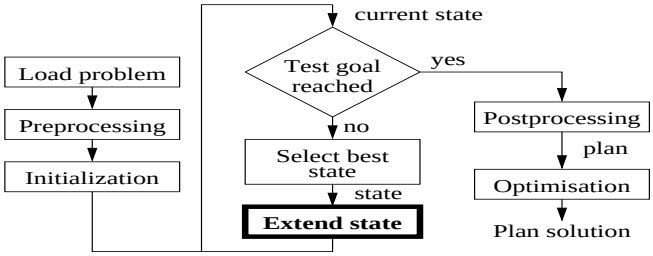

Figure 1: asyMov global search process

The aSyMov (a Symbolic Move3d) planner needs three input files: a geometric description of the environment, a PDDL2.1 domain and problem description and a description of the links between the symbolic and the geometric representations.

5.1 Global principles

aSyMov is a forward search planner in the state space (fig. 1). When the goal is reached, a step of post-processing is performed in order to extract, optimize and coordinate the trajectories using geometric tools. Before that it select a state in the front search (generally the last state reached). This step is important because the roadmap learning process can change the interest of the states, so backtrack in early stage can be useful. Then we will try to reach a new state from the selected state (fig 2).

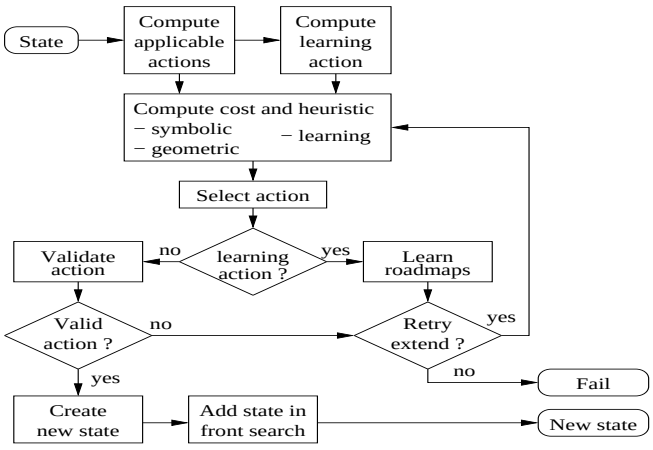

Figure 2: Extend state in search process

At each step, the planner selects applicable actions and computes costs and heuristics(§5.2). The costs of applicable actions can be realistically estimated, based on paths length provided by the motion planner. The cost of the learning action depends on the computation time taken by the corresponding roadmap extension functions and is based on the success rate for node sampling.

Action selection is performed using costs and heuristics. Moreover this selection takes into account a random part to

keep the probabilistic completeness property of the whole algorithm. At the beginning of the search process, when the roadmaps are empty, their heuristics and costs are very low, in order to favor an initial learning of the various roadmap topologies. If the selected action is not the learning action, and if it is valid, the planner adds a new state to the front search.

Even if we will not develop the algorithm of front search handler in this paper, we must underline one big difference with classical task planning: an action that is not currently applicable may become applicable after a roadmap extension. So, the front search must also attach to each state the list of non-currently applicable geometric actions. This list will re-consider if their corresponding connected components are modified (as a result of a learning action) making the state selectable again (fig 1).

5.2 A hybrid heuristic

The planner uses a hybrid heuristic that combines information from both task and motion planners.

The information used from the task planner is the estimation of the plan length to reach the goal from a given state (i.e. the symbolic state after the action application). Because our symbolic problems are simple we can use plans produced by a state of the art task planner (currently Metric-FF [Hoffmann, 2002]). For more complex description, we will use a relaxed problem like in the Metric-FF heuristic.

The information used from the motion planner is based on the roadmaps connectivity. For instance, if a final geometric state is connected to the current state, we can determine the minimal number of roadmaps that must be crossed to link the two states.

5.3 Learning

When the search starts, the roadmaps corresponding to the different states are "empty". We talk about learning because roadmap can be seen as learning structure of the environment which can be reuse for other problem instance. When a learning action is chosen, we must extend the roadmaps by the random sampling of new nodes. This sampling must let us gain a better knowledge of the environment in order to make an action choice at the next step. To do so, we have developed different extension strategies based on the task planner.

The first strategy consists in developing a symbolic plan from the current state to the goal. All geometric action involved in the plan need one or more roadmaps that are candidates to further extension. The second strategy gives the same weight to all actions that are applicable in the current state. In our current implementation, the strategy is chosen by the user. We will implement, in the future, a choice of the best adapted strategy depending on the context. For example, the first strategy seems more appropriate when the environment is not too constrained geometrically.

5.4 Validation and adaptation

In order to limit the construction of expensive roadmaps, we first consider each robot (or composed robot) as if it was alone in the static environment. It is the role of the validation step to test if there is a collision with the other objects or robots and to adapt the trajectories appropriately, if they exist.

This validation is based on the accessibility and the constraints of the geometric position (§4.3). When we need to change a position instantiation (i.e. a valid node), for example to reach a new node belonging to another roadmap, a new accessibility list is computed and used to choose a node that satisfies the new constraints. If no node is found, it is also possible to try to re-consider the old instantiation of the other robots and/or objects positions.

6 Examples and first results

The presented examples involve (fig. 3) a simple static environment composed of a wall and furniture. There are two identical robots: *forklift-1* and *forklift-2* initially situated on the left and the right sides. The problem is to bring the object named *flat-box* to its goal position (fig. 3 (a),(e)).

In problem 1 (fig. 3 (a)), the free path to the goal is blocked by another object named *big-box*. This variant belongs to the basic problems class. Its PDDL domain and problem are generated automatically from a meta-description. Depending on the learning steps, two solutions may be found by the planner: in the first case (fig. 3 (b)), *forklift-1* picks the *big-box* and puts it on an intermediate position, while *forklift-2* takes the *flat-box* and crosses the freed passage towards the goal. Following the progressive instantiation process described above, the final configuration for *big-box* is in fact changed when a new constraint is added when *forklift-2* wants to enter the other room. There is no way for forklift-2 to reach its desired position until the big-box accessibility, after a put action, is large enough to find a new configuration to avoid collision

In the second solution, *forklift-2* gives the *flat-box* to *forklift-1* through the "window" in the wall (d).

Problem 2 is a variant that involves a "purely" symbolic aspect: a *have-magnetic-key* predicate, two specific positions (*key-position* and *door-position*), a *get-the-key* action and an *open-door* action. The path to the goal is blocked by a door that must be opened. One solution can be simply (d). Another solution that involves forklift-1 which will get the key and then reach door-position in order to use it for opening the door (f): the path is freed for forklift-2 (g). This variant emphasizes the possible intricacies between geometrically constrained and purely symbolic actions.

To validate our approach, we compare the average results (cpu time on a 1.6 Ghz pentium 4) obtained with and without heuristic on ten resolutions without learning (i.e static roadmaps) and with a heuristic only based on the symbolic plan length.

problem	with heuristic nb step / time	without heuristic nb step / time	gain
1	388 / 2.4	1120 / 16.6	2.88 / 7
2	92 / 0.04	401 / 0.08	4.35 / 2

The second problem is more "symbolic" than the first one. As expected, the symbolic plan length based heuristic has a better gain in number of search step. The better gain in cpu time for the first one can be explain by the fact that the planner has to do more validation without heuristic and the validation is a costly process.

7 Conclusion and future work

We have presented an integrated planner that combines task and motion planning capabilities to solve realistic real world

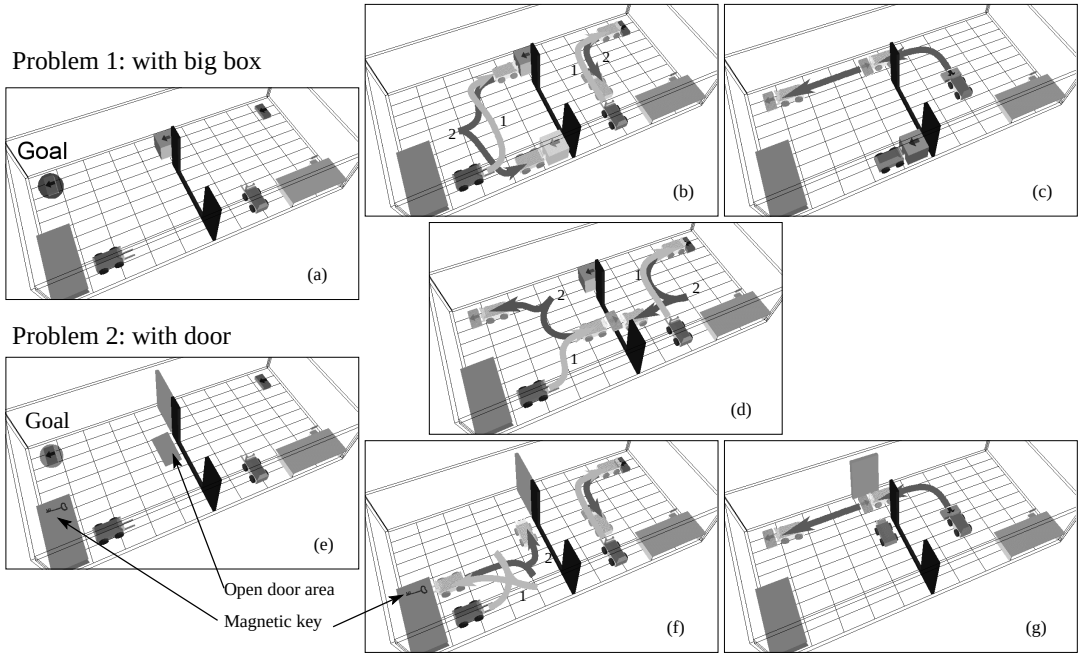

Figure 3: Illustrative examples. (a) Problem 1: move the flat box to its goal position. (e) Problem 2: same goal but a locked door and a key are added. (b) (c) Show a solution to Problem 1: move the big box and then carry the flat-box to its goal. (d) Solution common to both problems, pass the flat box through the "window". (f) (g) Solution to Problem 2: take the key, open the door and then carry the flat-box.

manipulation problems. Indeed, in such applications symbolic and geometric constraints are often intricately linked and cannot be treated by a two step hierarchical system.

The implementation is near to be finished, only the learning part is in construction. The first result are encouraging. It is able to deal with "simple" problems (but not resolved in the current state of the art).

The next step is to investigate means to construct a more efficient heuristics. This will be done to allow the planner to face more complex tasks and environments.

aSyMov handles a real world model which takes into account geometric, topologic and relation data. The toplogy is catch by the PRMs and the relation on "things" which not are necessary places are represented by predicates. Thus it provides plans with a rich expressivness. The plans point out not only an actions series but also a physical way to realize them. To apply such type of plan in a robotic application, it will be necessary to use additional specific systems to further resolve some tasks like manipulation at a lower level.

References

[Ahuactzin *et al.*, 1998] J.-M. Ahuactzin, K. Gupta & E. Mazer. *Manipulation Task Planning for Redundant Robots: A Practical Approach.* In K. Gubta & A.P. Del Pobil, editors, Practical Motion Planning in Robotics: Current Approaches and Future Directions, pages 79–113. J. Wiley, 1998.

[Alami *et al.*, 1994] R. Alami, J.P. Laumond and T. Siméon. *Two manipulation planning algorithms*. In *Algorithmic Foundations of Robotics (WAFR94)*, K. Goldberg et al (Eds), AK Peters, 1995.

[Fox and Long, 2001] M. Fox and D. Long *PDDL2.1: An extension to PDDL for expressing temporal planning domains*. Technical Report, Univ. of Durham, UK, 2001.

[Gravot *et al.*, 2002] F. Gravot, R. Alami and T. Siméon *Playing with Several Roadmaps to Solve Manipulation Problems*. In *IEEE International Conference on Intelligent Robot & System*, Lausanne, September, 2002

[Hoffmann, 2002] J. Hoffmann *Extending FF to Numerical State Variables*. In *Proceedings of the 15th European Conference on Artificial Intelligence*, Lyon, France, July 2002.

[Kavraki and Latombe, 1994] L. Kavraki and J.C. Latombe *Randomized preprocessing of configuration space for fast path planning*. In *IEEE International Conference on Robotics & Automation*, San Diego, May, 2002

[Koga and Latombe, 1994] Y. Koga & J.-C. Latombe. *On Multi-Arm Manipulation Planning. IEEE International Conference on Robotics and Automation*, vol. 2, pages 945–952, 1994.

[Lozano-Prez *et al.*, 1987] T. Lozano-Prez, J.L. Jones, E. Mazer & P.A. O'Donnell. *HANDEY: A Robot System that Recognizes, Plans, and Manipulates.* Proceedings of 1987 *IEEE International Conference on Robotics and Automation*, March 1987.

[Sahbani *et al.*, 2002] A. Sahbani, J. Cortès and T. Siméon *A Probabilistic Algorithm for Manipulation Planning under Continuous Grasps and Placements*. In *IEEE International Conference on Intelligent Robot & System*, Lausanne, September, 2002

[Siméon *et al.*, 2000] T. Siméon, J-P. Laumond, C. Nissoux. Visibility based probabilistic roadmaps for motion planning. *Advanced Robotics Journal*, Vol. 14, n^o 6, 2000.