A Robot Task Planner that Merges Symbolic and Geometric Reasoning

Stéphane Cambon, Fabien Gravot and Rachid Alami
LAAS-CNRS
7, avenue du colonel Roche
31077 Toulouse cedex 4
{scambon,fgravot,Rachid.Alami}@laas.fr

Abstract.
We have developed an original planner, aSyMov, that has been specially designed to address intricate robot planning problems where geometric constraints cannot be simply "abstracted" in a way that has no influence on the symbolic plan. This paper presents the ingredients that allowed us to establish an effective link between the representations used by a symbolic task planner and the representations used by a realistic motion and manipulation planning library. The architecture and the main plan search strategies are presented together with an illustrative example solved by a prototype implementation of aSyMov. At each step of the planning process both symbolic and geometric constraints are considered. Besides, the planning process tries to arbitrate between finding a plan with the level of knowledge it has already acquired, or "investing" more in a deeper knowledge of the topology of the different configuration spaces it manipulates.

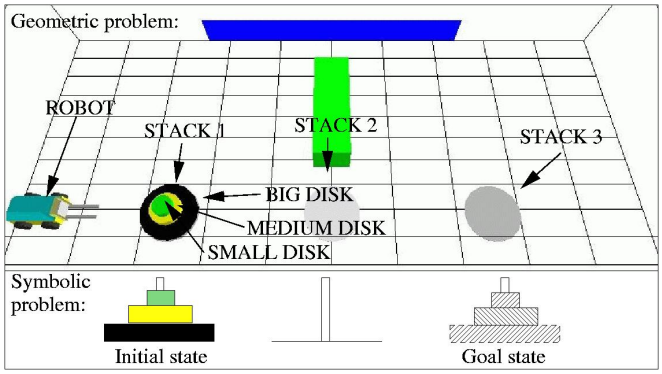

Figure 1. GHTP-3: An instance of the the Geometric Hanoi Tower Problem: 3 cylinders manipulated by a non-holonomic fork-lift in a geometrically constrained environment

1 Introduction

The domain independent planners have been improved dramatically and can solve more and more complex problems. However, their effective use in robotics is still very limited. One main reason, in our point of view, is the gap between the representation they are based on and the physical world. For example, depending on the context (size and shape of the robot and the object, environment configuration...), when a robot grasps an object the shape of the "composed" robot, i.e. the robot and the attached object, may have drastic consequences on its future actions or actions of other robots. This is generally ignored by the symbolic task planners.

In this paper, we assume that, besides a standard symbolic description, we have a complete description of the geometry of the environment. Figure 1 illustrates such data. In order to exhibit clearly the main topics, we have chosen an example based on the "classical" Hanoi Tower Problem. We call it the "Geometric Hanoi Tower Problem with 3 disks" (GHTP-3). In our example, the tower is composed of cylinders (called "disks") that can be moved by a non-holonomic robotized fork-lift.

Depending on the size of the obstacle in the middle of the environment this problem can become unsolvable by a hierarchical planning approach where first a symbolic plan is found and then the robot motions are planned. Indeed, a good symbolic plan, with an optimal number of actions, will manage to build on `stack-2` a two-disks tower, then move the big disk to `stack-3`. In the instance of GHTP-3 shown on figure 1, there is no possible path for the robot to transfer the Big Disk directly from `stack-1` to `stack-3` if a disk is placed on `stack-2`. Indeed, the obstacle in the middle leaves enough room for a robot transporting a small or medium size disk to travel between `stack-1` and `stack-3` even if `stack-2` is occupied (figure 2). aSyMov (a Symbolic Move3d [13]) is able to detect such a situation and to find a feasible plan (figure 5). Our approach is able to take into account not only discrete but continuous sets of grasps and placements. However, for the sake of clarity we will limit ourselves here to GHTP-3, where we do not allow the disks to be placed arbitrarily in the environment. However, we allow infinitely many grasp configurations: any contact position of the robot in a radial orientation relatively to a disk will be considered as a possible grasp position.

All robot motions (trajectories, reachable grasp configurations..) depend not only on the static environment but on the context (i.e. the

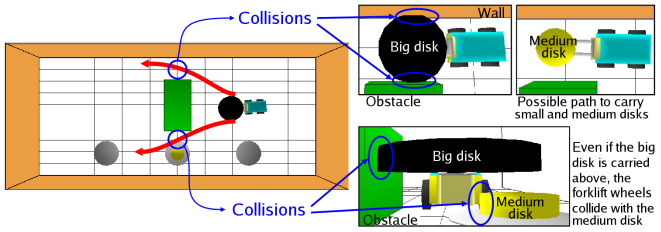

Figure 2. Geometric constraints: the obstacle forbids the forklift to carry the Big Disk between stack-1 and stack-3 if the medium disk is on stack-2. For the smaller disks it is possible to find a path.

positions of all the other objects and robots). Each context may possibly create a different free-space topology and consequently create specific constraints on robot motions. We may have a combinatorial explosion of contexts particularly if we consider a continuous set (or a great number of discrete positions) of placements for the objects.

When the number of such contexts is small, it can be taken into account by the programmer. This is typically done in problems when the topology of the environment is provided at the symbolic level (the `CONNECTED` predicate is often used) and where situations that entail a change in the topology (adding/deleting `CONNECTED` facts) are predefined.

The only systematic way to guarantee the separation between the symbolic and the geometric aspects (without loss of solutions) would consist in pre-computing beforehand the topology of the free-space for each context and analyzing for each one the feasibility of all actions. This is clearly not reachable. aSyMov is a tentative answer to this problem.

We will discuss in the sequel the ingredients that allowed us to establish an effective link between the representations used by a symbolic task planner and the representations used by a realistic motion and manipulation planning library. We then describe the architecture and the main plan search strategies together with an illustrative example solved by a prototype implementation of our planner. At each step of the planning process both symbolic and geometric constraints are considered. Besides, the planning process tries to arbitrate between finding a plan with the level of knowledge it has already acquired, or "investing" more in a deeper knowledge of the topology of the different configuration spaces it manipulates.

2 Geometrical model and algorithms

In order to tackle realistic problems, we rely on advanced geometric planning techniques (Probabilistic Roadmap Methods (PRMs) [8, 14] and their use for the so-called manipulation planning problem [1, 12, 5].

2.1 Some definitions

Environment: We work in a 3D-world. It consists of static components (obstacles, walls, bookshelves ...) and dynamic components: robots and "movable" objects.

Roadmap: In order to capture the topology of the free space, the motion planner computes randomly a graph called roadmap. A node represents a collision-free configuration (one instance of the degree of freedom of the mechanical system) and the edges are valid robot motions. A roadmap can be composed of several connected components. In each one, there is a valid path connecting any two nodes in the considered free space.

Transit/Transfer/Placement: A manipulation problem [1, 9] involves two types of motions, the *transfer* of an object taken by a robot and the *transit* motion of a robot between two grasping positions. These motions can be represented by roadmaps. The set of all stable object positions is called *placement*. It can also be represented by a roadmap. A solution to a manipulation problem is a sequence of transit and transfer motions.

2.2 Specificities of the multi-robot manipulation problems

One main feature in our approach is the notion of "robot composition" and the use of several specialized roadmaps [5]. 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 some legs for assembly problems, and a robot carrying an object is the composition of the robot and the object. With this definition, a transfer motion is a valid motion for the composed robot and we can build a roadmap for it. Consequently, for a multi-robot manipulation planning problem we define several specialized roadmaps for each type of motion. Naturally there are connections between these roadmaps (see fig 3). Such connections correspond to robot composition. For example, the nodes in a *transfer* roadmap that correspond to a stable object position can be linked to a node in the *placement* roadmap and to a node in the robot *transit* roadmap.

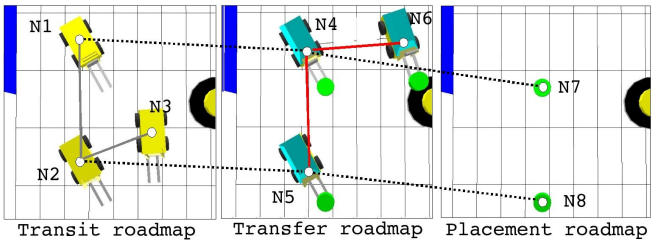

Figure 3. Illustration of connections between roadmaps (in discontinuous lines). The node N4 of the transfer roadmap is the concatenation of node N1 of the transit roadmap with node N7 of the placement roadmap. N5 is the concatenation of N2 and N8.

For manipulation problems involving several robots and objects, the number of possible roadmaps corresponding to all possible compositions may grow rapidly. We need a strategy to choose the roadmap we have to expand or explore. We claim that a symbolic level can guide this search process. 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 the object roadmaps, and no expensive search will be performed in the other robots' roadmaps. The next section shows how we link the roadmaps representation with a symbolic description.

3 Topologies and symbolic representation

The first motivation to have a symbolic representation is to guide the roadmaps expansion and exploration. Our symbolic representation, coded into a task planner, will be a relaxed instance of the problem in which we will consider that it is always possible to find a motion without collision between two nodes in every roadmap.

The second motivation is to enrich the robot manipulation problem with symbolic constraints. For instance, in the GHTP-3 a given disk cannot be put on a smaller one.

3.1 Symbolic representation of topologies

A *symbolic position* of a robot or a movable object is a term representing a subset of the configuration nodes of one roadmap that satisfy a given property. We have defined a nomenclature for such terms: `P_{robot name}_{roadmap name}_{property}`

For instance, the configurations where a robot `R` can grasp an object `O` will be denoted by `P_R_TI_TA_O`. Indeed this term represents all the roadmap nodes which (1) are in the transit roadmap (`TI`) of the robot `R` and (2) can be composed with a node from the placement roadmap of `O` to form a third node which is in the transfer (`TA`) roadmap of the composition `R-O`. On figure 3, `P_R_TI_TA_O` represents the set of node $\{N1, N2\}$. `P_R-O_TA_TI` represents all the

nodes of the transfer roadmap (TA) of R-O which are a composition of a node from the transit roadmap (TI) of R and a node from the placement roadmap of O. On figure 3, P_R-O_TA_TI represents the set $\{N4, N5\}$.

Another example: the term P_R_TI_RELOAD_BATTERY represents all the nodes of the transit (TI) roadmap of R where the planner can apply a RELOAD_BATTERY action (see §3.2).

This splitting by properties of the global topology allow us to manipulate a constant and given set of terms that can be computed automatically to satisfy a given constraint: ("R can grasp O; R can RELOAD_BATTERY; ...").

3.2 Symbolic domain construction

This representation choice brings us naturally to specify a set of *basic* types and predicates for what we call *the basic problem* (a pick-and-place problem). We have chosen PDDL2.1 [4] for this specification. There are three main types of symbolic terms: *robot*, *movable_object* and *symbolic_position*. On these types we construct a *basic* predicate set:

- **(composed ?r1 ?r2 ?r3)**: the composition of two robots or movable objects ?r1 and ?r2 is possible and forms a third robot or movable object ?r3. (eg: (COMPOSED R O R-O))
- **(belongs-to ?p ?r ?roadmap-type)**: a symbolic position ?p belongs to the roadmap of type ?roadmap-type of a robot or a movable object ?r. (eg: (BELONGS-TO P_R_TI_TA_O R TI))
- **(has-purpose ?p ?pos-type)**: is used to associate a symbolic meaning ?pos-type to a geometric position. Such a property is necessary when a robot has to reach a given position as a pre-condition to apply a "purely symbolic" action (i.e. an action that may have geometric constraints preconditions but no geometric consequences). For instance, the action (RELOAD_BATTERY ?robot ?p) would have as a precondition the predicate (ON ?robot ?p) and (HAS_PURPOSE ?p RELOAD_POSITION). We use also this predicate to specify the initial and goal symbolic positions.
- **(connection ?p1 ?p2)**: denotes that it is possible to find a connection between two symbolic positions ?p1 and ?p2 which do not belong to the same roadmap. (eg: (CONNECTION P_R_TI_TA_O P_R-O_TA_TI))
- **(on ?r ?p)**: the robot or movable object ?r is located at the symbolic position ?p. Moreover ?r is active, indeed a robot or a movable object can "disappear" if it is composed with another one or decomposed (see the effects of the STACK_ON action).

In the *basic problem*, only the *"on"* predicate will be fluent. In fact, the instances of the *composed*, *belongs-to*, *has-purpose* and *connection* are fixed and provide a description of the set of available roadmaps and their links. This is, as mentioned above, a relaxed description of the real structure. It is the role of the geometric level to compute a more refined topology.

Only a few predicates are specifically defined in our planner. We can combine them to build the appropriate actions (that have geometric conditions and effects) and also add "pure" symbolic ones. Hereafter, we give the STACK_ON action used for GHTP-3.

```
(:action STACK_ON
 :parameters
     (?r_plus_disk  - robot  ?p3 - position
      ?r - robot  ?p1 - position
      ?disk1 - obj ?p2 - position
      ?stack - position_type ?disk2 - obj
      ?height ?height_plus - position_type)
 :precondition (and
      ;; relaxed representation of geometry
      (composed ?r_plus_disk ?r ?disk1)
      (on ?r_plus_disk ?p3)
      (belongs-to ?p3 ?r_plus_disk TA)
      (belongs-to ?p1 ?r TI)
      (belongs-to ?p2 ?disk1 PL)
      (connection ?p3 ?p1)
      (connection ?p3 ?p2)
      ;; constraints
      (has_purpose ?p2 ?height_plus)
      (has_purpose ?p2 ?stack)
      ;; purely symbolic part
      (bigger_than ?d2 ?d1)
      (on_top ?stack ?d2)
      (height ?stack ?height)
      (minus_height ?height_plus ?height))

 :effect (and
      ;; geometric effects
      (not (on ?r_plus_disk ?p3))
      (on ?r ?p1)
      (on ?disk1 ?p2)
      ;; symbolic effects
      (not (on_top ?stack ?disk2))
      (on_top ?stack ?disk1)
      (not (height ?stack ?height))
      (height ?stack ?height_plus))
)
```

4 Algorithms and strategies

aSyMov needs three input databases: (1) a geometric description of the environment, (2) a PDDL2.1 domain and problem description and (3) a description of the relations between the geometric objects described in (1) and the terms described in (2). It is possible and easy to change the symbolic and/or the geometric constraints by editing their corresponding files. For instance, in the GHTP-3, we can change the stacking rules in the symbolic part without modifying the other files or symmetrically, change the obstacle or robots dimensions in the geometric part.

4.1 Global principles

aSyMov is a A*-like forward search planner in the state space. This kind of search allows us to maintain an explicit description including all configurations of robots and objects. This is essential in order to compute robot trajectories. The planner can start with a set of empty roadmaps that it will explore incrementally during its search. Such roadmaps can be re-used for subsequent planner calls. At each step it will have to choose between trying to find a plan with the level of knowledge it already has acquired (the roadmaps as they are), or to "invest" more in a deeper knowledge of the topology of the different configuration spaces that it manipulates (selection and expansion of some roadmaps).

Note that an action that has not been validated previously may become valid after a roadmap expansion. Consequently, the front search must also attach to each state the list of applicable but non-currently valid geometric actions. This list must be re-considered whenever the roadmaps have been updated by a roadmap expansion. So we cannot remove simply a state from the front search. We stop the search after a predefined number of steps of the core procedure which may add a new state to the front search: the *extend state* algorithm (fig 4).

The first operation of this procedure is to compute the *applicable* actions on the basis of the symbolic part of the state. Thus, we ensure the satisfaction of the symbolic constraints. We introduce complementary actions called *roadmap-expansion* which implement incremental roadmaps expansion.

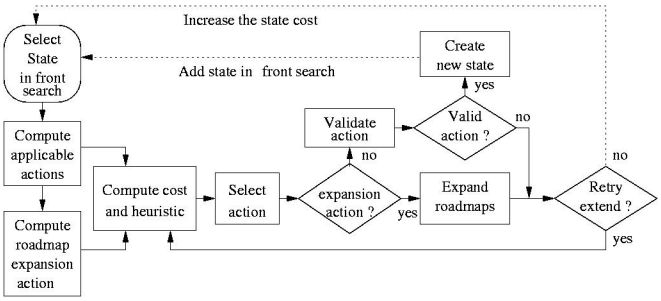

Figure 4. Algorithm to extend a state from the front search

The second operation computes costs and heuristics of the actions. The heuristics for *applicable* actions are based on the length of plans found by a task planner, currently Metric-FF [7], on our symbolic description. This value is modified by taking into account the connectivity of the roadmaps corresponding to the actions involved in the plan. If the roadmaps connectivity is not sufficient the cost is increased. The heuristics for the *roadmaps-expansion* actions depends on the number of failures of previously selected *applicable* actions.

The selection includes a random part. Even if the best action (i.e. for a symbolic planner) is often the most likely, all feasible actions have at least a small probability to be chosen. If the symbolic planner is complete and the roadmap expansion has the probabilistic completeness property, our planner keeps this probabilistic completeness property. The non-existence of solution is so unprovable.

If the selected action is a *roadmap-expansion*, the planner selects and expands a set of roadmaps that are suggested by the solution to the relaxed problem. If the selected action is an *applicable* one and if it has geometric consequences (action which contains "on" predicates in its effects), the planner tries to validate it (see 4.2).

At the end of the *extend state* procedure, the planner recomputes the cost of some states of the front search. If the procedure fails to extend a state, its cost is increased.

When the goal is reached a set of post-processing steps (not detailed here) can be performed in order to clean the plan (redundant actions can appear), to detect actions that can be performed in parallel [11], to smooth the robot trajectories and to synchronize multi-robot motions.

4.2 The validation process

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 process to plan paths, to test if there are collisions with the other robots or movable objects and to adapt the trajectories appropriately, if they exist.

This process can be seen as an incremental instantiation process of the symbolic positions. When we try to validate an applicable action which implies a motion, we try to find a valid roadmap node for this position. We stop the process when at least one valid node is found but we keep all the other candidate nodes. With this mechanism, we can re-instantiate the position if the current instance prevents further motions. For instance, if a previous action has moved an object `O` to a position that has been instantiated with a node $n1$ and if this object obstructs the motion that is currently in the validation process, the planner can try to validate another node $n2$ for `O`.

The validation process is a costly operation, which may, in the worst case, be exponential in the number of robots/object. We explain how we try to reduce as much as possible the computation costs by using a lazy evaluation strategy in [6]. Let us simply mention here that the validation process does not change the symbolic part of the plan under construction. It is not a simple validation stage for one action as it can be performed with a simple hierarchy. It is able to back-propagate geometric constraints on all the plan under construction.

It is important to note that the planner will explore intensively the state-space only when it faces intricate validation situations. We have tried to give an estimate of this exploration, called "connectivities explored" in the next section.

5 Results on the GHTP-3

We have already validated in [3] the usefulness of the symbolic problem as a heuristic guide even for the *basic problem*. We present here below average results on 50 runs (table 1) obtained on several GHTP-3 instances[1]:

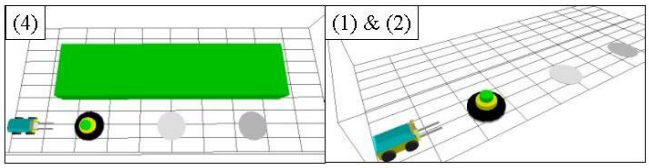

Figure 6. Illustration of the GHTP-3 instances (1) (2) and (4).

- **instance (1): without obstacles and with precomputed roadmaps**: The same problem shown in fig 1 but without any obstacle (fig 6). Moreover, we switch off the roadmap expansion actions and use precomputed roadmaps. In this case, the plan proposed by a sequence of a conventional task planner and a motion planner is the same than the plan found by aSyMov. In such a case, of course, the heuristic plan provides a good guidance.
- **instance (2): without obstacles:** The search is started with empty roadmaps.
- **instance (3): with obstacle:** This is the problem shown in fig 1. The obstacle is such that it obstructs the direct transport of the big disk from stack-1 to stack-3. As already mentioned and illustrated, aSyMov is able to find a valid plan (fig. 5) that satisfies both geometric and symbolic constraints.
- **instance (4): with a "difficult" obstacle**: The shape and size of the obstacle (fig 6) has been chosen so that it leads to really constrained motion planning for a non-holonomic robot carrying the Big Disk from stack-1 to stack-3 while a disk is placed on stack-2. In this case 76% of the plans found are similar to those found for instance 3 and 24% are similar to plans found for instances 1 or 2.

	CPU time	Nb step	Connectivities	Plan length
(1)	0.266	38.5	17.5	28
(2)	9.97	63.7	25.4	28.8
(3)	271.6	243.8	71.2	44
(4)	298.9	213.6	62.8	40.16

Table 1. Results on several instances of GHTP-3.

- **CPU time**: as measured a 3.2 GHz Pentium 4 HT.
- **Nb step**: number of calls of the *extend state* algorithm.
- **Connectivities explored**: In task planning for robotics domain, the environment connectivity is often expressed by a `CONNECTED` predicate (e.g. (`CONNECTED` position1 position2)). However, to reflect correctly the reality this predicate should take into account the other robots and movable objects positions (e.g. (`CONNECTED` robot1 position1 position2) with `object2` on `position3`,

[1] The interested reader may find at http://www.laas.fr/~scambon a number of mpeg video segments that illustrate the plans produced by aSyMov.

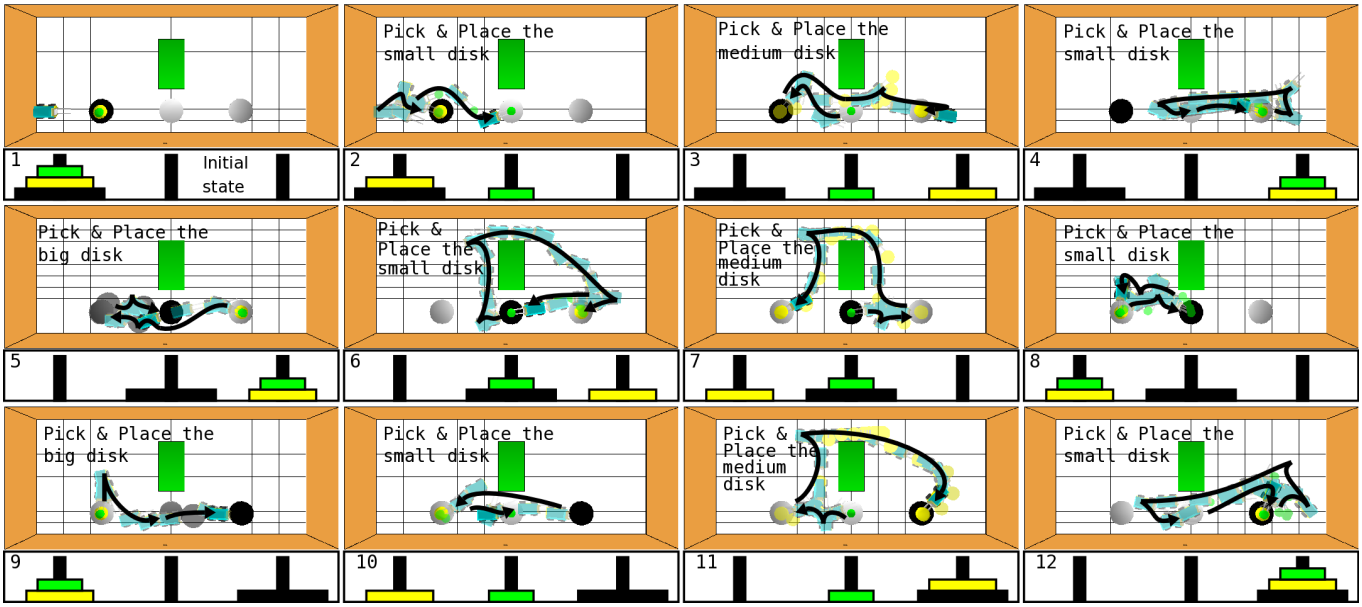

Figure 5. A plan found for the instance 3 of the GHTP-3: actions and corresponding paths. This is the shortest possible plan (in number of action) because there is no possible path for the robot to transfer the Big Disk directly from `stack-1` to `stack-3` if a disk is placed on `stack-2`.

`robot3` on `position4`...). For the instances of the GHTP-3 presented here, the number of possible `CONNECTED` facts is 151[2] but their boolean values are obviously unknown in advance by aSyMov (and considered as TRUE in the relaxed problem). We give here the number of connectivities that have been effectively explored during the plan search. The minimum is 14 (e.g. number of necessary trajectories) for instances (1) and (2) and 22 for instance (3).

- **plan length**: We estimate that the optimal plan length for instances (1) and (2) is 28, 44 for (3) and we note that the optimal plan for (4) is 28.

The performance is strongly dependent on the difficulty of the geometric problem, motion planning for non-holonomic robots in constrained environment add many difficulties. This correlation appears also with the *connectivities* measure: when the planner faces difficulties it tries other ways.

6 Conclusion and future work

To the best of our knowledge, aSyMov is the first task planner that deals with a complex geometric representation involving a manipulation context. In a motion planning point of view, we solve relatively complex multi-robot manipulation problems. The search is guided thanks to the use of symbolic representation which let us compute a relaxed solution. In a task planning point of view, our system is the only planner which takes into account a precise description of the world geometry and the effective ability of the robots.

With a robotics application, our planner has two main advantages over other planners. First, the plans have better chance to be valid, because they take into account the geometric consequences of robots' actions. Second, the plans have a good expressiveness, they provide not only the actions set but also the way to achieve them.

Our main point here is that we have developed an effective link between symbolic and motion planning and a number of mechanisms for guiding the search that allow to deal with a new class of intricate robotics problems.

Our future investigations will involve a more elaborate control that can take into account more information produced by the validation and the roadmap expansion activities. Another aspect involves the choice of a symbolic planner that allows to integrate domain-specific control strategies induced by the manipulation context. A planner like SHOP [10] or TLPlan [2] could be good candidate.

[2] enumerated by hand following all possible motions in all possible symbolic contexts

REFERENCES

[1] R. Alami, J.-P. Laumond, and T. Siméon, 'Two manipulation planning algorithms', *WAFR94*, (1995).
[2] F. Bacchus and F. Kabanza, 'Using temporal logics to express search control knowledge for planning', *Artificial Intelligence, vol 116*, (2000).
[3] S. Cambon, F. Gravot, and R. Alami, 'Overview of aSyMov: Integrating motion, manipulation and task planning', *ICAPS Doctoral Consortium*, (2003).
[4] M. Fox and D. Long, 'PDDL2.1: An extension to PDDL for expressing temporal planning domains', *Technical Report, Univ. of Durham, UK*, (2001).
[5] F. Gravot, R. Alami, and T. Siméon, 'Playing with several roadmaps to solve manipulation problems', *IROS*, (2002).
[6] F. Gravot, S. Cambon, and R. Alami, 'aSyMov: a planner that deals with intricate symbolic and geometric problems', *ISRR*, (2003).
[7] J. Hoffmann, 'Extending FF to numerical state variables', *ECAI*, (2002).
[8] L. Kavraki and J.C. Latombe, 'Randomized preprocessing of configuration space for fast path planning', *ICRA*, (2002).
[9] Y. Koga and J.C. Latombe, 'On multi-arm manipulation planning', *ICRA*, (1994).
[10] D. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila, 'SHOP: Simple hierarchical ordered planner', *IJCAI*, (1999).
[11] P. Régnier and B. Fade, 'Complete determination of parallel actions and temporal optimization in linear plans of actions', *European WorkShop on Planning*, (1991).
[12] A. Sahbani, J. Cortès, and T. Siméon, 'A probabilistic algorithm for manipulation planning under continuous grasps and placements', *IROS*, (2002).
[13] T. Siméon, J.-P. Laumond, and F. Lamiraux, 'Move3d: a generic platform for path planning', *4th International Symposium on Assembly and Task Planning*, (2001).
[14] T. Siméon, J.-P. Laumond, and C. Nissoux, 'Visibility based probabilistic roadmaps for motion planning', *Advanced Robotics Journal*, (2000).