aSyMov: A Planner That Deals with Intricate Symbolic and Geometric Problems

Fabien Gravot, Stéphane Cambon, Rachid Alami

aSyMov: a planner that deals with intricate symbolic and geometric problems

Fabien Gravot[1], Stephane Cambon[1], and Rachid Alami[1]

LAAS-CNRS
fgravot, scambon, Rachid.Alami@laas.fr

Abstract. We propose an original approach to integrate symbolic task planning and geometric motion and manipulation planning. We focus more particularly on one key aspect: the relation between the symbolic positions and their geometric counterparts. Indeed, we have developed an instantiation process that is able to propagate incrementally task-dependent as well as 3D environment-dependent constraints and to guide efficiently the search until valid geometric configurations are found that satisfy the plan at both levels. The overall process is discussed and illustrated through an implemented example.

1 Introduction

In the last years, task planners have been improved to solve more and more complex symbolic problems. However, the difficulty to successfully apply such planners to robotics problems still remains. This is due to 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 it future actions or actions of other robots. This is generally ignored by symbolic task planner.

On the other hand, path planners are dedicated to geometric problems. State of the art motion planning systems can even plan for manipulation tasks [4],[10],[1]: they handle complex geometric constraints and for instance they can compute the intermediate Pick&Place actions which are sometimes needed for re-grasping objects. However such planners cannot handle symbolic relations in a generic way.

aSyMov has been specially designed to address robot planning problems where geometric constraints cannot be simply "abstracted" in a way that has no influence on the obtained plan. aSyMov is an original planner (a Symbolic Move3d)[6] which uses in a hybrid way the competence of a task planner (Metric-FF [8]) and a path and manipulation planner (Move3d[12,5]). At each step of the planning process both symbolic and geometric data are considered.

The symbolic planner deals with "symbolic positions" which are considered as the specification of a sub-space of the global C-space. Move3d is invoked to build incrementally various roadmaps that will be used to select robots and objects configurations and to propagate geometric constraints. Indeed, one of the main ideas of aSyMov is to allow flexibility in the search for the configurations associated with a symbolic position. This is represented by an "accessibility list" whose elements

belong to roadmaps built by Move3D. As we will see, the management of accessibility lists allows to back-propagate new geometric constraints on previous symbolic positions.

After a brief description of aSyMov and of the representation that allows to associate the symbolic and geometric part of the planner, we will focus on the main contribution of this paper: the management of accessibility lists and on the back-propagation process. The overall process is discussed and illustrated through an implemented example.

2 aSyMov: Symbolic and Motion Planning

2.1 Geometric Manipulation Planning

The motion planning library that we use is based on Move3D[12]. It relies on *Probabilistic Roadmap Methods* (PRMs) which have proved to be efficient for highly dimensional motion planning problems [9]. A roadmap is built by picking nodes and linking them to previous connected components. Once the initial and final roadmap nodes belong to the same component of the roadmap, a solution is found.

Manipulation planning problems are a first step toward geometric models of actions. Indeed, besides robots and obstacles, they involve (movable) objects and, consequently, two types of motions [4]: *transfer* motions of objects taken by a robot and *transit* motions of robots between two grasping positions. A solution to a manipulation problem is a sequence of transit and transfer motions that are performed in different roadmaps corresponding to a robot alone or to a robot that grasps an object. Recent results [11] have elaborated methods that explore a sub-space called $Grasp \cap Placement$ and to solve efficiently complex "one robot one movable object" manipulation tasks.

Our geometric planner[5] is an extension of this work. One main feature in our approach is the notion of "robot composition" and the use of several specialized roadmaps. 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 legs for assembly problems. A robot carrying an object is considered as a new robot that results from the composition of the robot and the object. With this definition, a transfer motion is a valid motion for the composed robot. For multi-robot manipulation planning problems we define several specialized roadmaps for each type of motion. Naturally there are connections between these roadmaps. Such connections correspond to robot composition. For example, a roadmap node in $Grasp \cap Placement$ roadmap can be divided into a roadmap node for the object roadmap and a roadmap node for the robot alone (*transit*), it can lead to a roadmap node for the *transfer* roadmap.

This will have a strong influence on the definition of the predicates for the task planner described below.

2.2 A Symbolic Representation and its Geometric Counterpart

We have chosen to represent the location of robots and object by a symbolic type called "position". It corresponds to a set of configurations. For instance, the configurations where a robot R can grasp an object O will be denoted by P_TI_TA_O_R. Indeed such a symbolic position corresponds to a transition between a transit motion (TI) and a transfer motion (TA). As a consequence of the grasp action, a a new robot R-O will be created. It may place (transition between TA and TI) the object at a position denoted by P_TA_TI_R-O

In order to tackle properly the interaction between the symbolic and the geometric aspects of robot problems, we have to define a framework where geometric consequences of symbolic actions can be expressed. We introduce the "*basic problem*" notion: it is a pure motion and manipulation problem. It can be very complex and intricate when, for instance, it entails the rearrangement of a number big boxes by a number of robots in a constrained room.

We use three main types of symbolic parameters: robot, movable object (a particular type of robot) and position. The following predicates are defined.

- **compose ?r1 ?r2 ?r3**: the composition of two robots ?r1 and ?r2 is possible and the result is a third robot ?r3. (eg: (COMPOSE R O R-O))
- **belongs-to ?p ?r ?roadmap-type**: a position ?p belongs to a roadmap of type ?roadmap-type for a robot ?r; examples of roadmap-types are TI (transit) and TA (transfer). (eg: (BELONGS-TO P_TI_TA_O_R R TI))
- **is-specific-pos ?p ?pos-type**: is used to declare that a position ?p is dedicated to a special treatment. For example, initial and goal position can be a ssociated to a unique geometric position. This predicate is also used to specify areas in which a robot must be located to apply a pure symbolic action.
- **connection ?p1 ?p2**: denotes that it is possible to find a connection between two positions ?p and ?p2 which do not belong to the same roadmap. (eg: (CONNECTION P_TI_TA_O_R P_TA_TI_R-O))
- **on ?r ?p**: robot ?r is situated at the symbolic position ?p.

With this set of predicates, and for the basic problem we can specify actions which add or remove "*on*" predicates. To stick with classical problems, we specify three main actions: *goto* (motion), *grasp* (composition), *ungrasp* (decomposition). Note that these actions are not built-in the planner but are simply specified thanks to the predicates described above. Here is the grasp action:

```
(:action grasp
 :parameters (?r    - robot              ?p1 - position
              ?o    - (either obj robot) ?p2 - position
              ?newrobot - robot          ?p3 - position)
 :precondition (and (on ?r ?p1)
                    (on ?o ?p2)
                    (belongs-to ?p3 ?newrobot TA)
                    (compose-robot ?newr ?r ?o)
```

```
                    (connection ?p1 ?p3)
                    (connection ?p2 ?p3))
 :effect (and   (not (on ?r ?p1))
                (not (on ?o ?p2))
                (on ?newr ?p3)))
```

Other “purely 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 for an *open-door* action.

2.3 A hybrid planning process

aSyMov is a forward search planner in the state space. Once the goal is reached, a post-processing is performed to extract, optimize and coordinate all the geometric trajectories thanks to dedicated tools.

The search strategy is close to a hill climbing [7] process. When a backtrack is triggered the next state to explore is selected from the complete front search according to a heuristic function. It is important to note that the heuristic estimator we consider is dynamic: the extension of the roadmaps that may happen during the search might lead to revise the estimated interest of some states. Figure 1 presents the core procedure to move from one state to another in the search space.

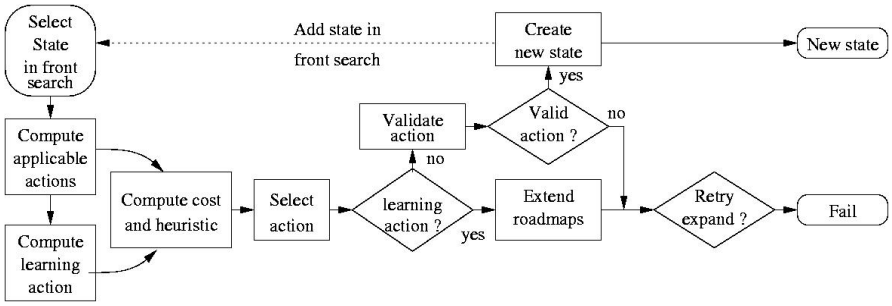

Fig. 1. State-space expansion in the search process

At each step, the planner selects applicable actions and computes costs and heuristics. The heuristics are computed on the basis of a symbolic plan. An applicable action which brings the symbolic state nearer to the symbolic goal will have more chance to be selected. In fact the symbolic level solves a relaxed version of the problem in which all paths are considered as valid. In the same time, the process can also decide to invoke “learning actions” i.e. actions that will cause a further exploration of the free-space of a given manifold through an extension of the available roadmaps. With this mechanism, the planning process can estimate computing costs and decide (1) to try to find a plan with the level of knowledge it already has, or (2) to “invest” more in a deeper knowledge of the topology of the different configuration spaces it manipulates.

We focus, in the sequel, on the sub-system named *"validate action"* (fig. 1). This module is in charge of deciding whether a given symbolic action assumed to be applicable by the symbolic planner, can actually be applied in the 3D world.

3 State and Positions

An illustrative example: In this example there are two robots fork-lifts called `F1` and `F2` and two movable objects (Flat box denoted by `FB` and Big box denoted by `BB`). For the sake of simplicity, our example does not involve "purely" symbolic actions. The goal is to carry `FB` from one room to the other room.

There are several ways to solve this problem. One of the solutions found by our planner (fig. 2) is to displace `BB` in order to clear the way to a robot to carry `FB` to its final position. Besides, the environment boundaries prevent robot `F2` to perform this last operation.

We will not discuss how the planner chooses the succession of actions given as solution. Instead we will explain how the accessibility lists and the geometric states are handled.

State Representation: The state can be divided into three parts. The first one is purely symbolic; it consists of all the symbolic predicates that have no relation with geometric aspects (not defined in §2.2).

The second part is the interface between the task planner and the motion planner. It is composed of the `(on ?r ?p)` predicates which describe the symbolic position of the robots and the objects. A geometric position is associated to each symbolic position: it stores the possible configurations of the robot (or object).

The third part is called *geometric state*. It is a set of configuration combinations (one per geometric position) that can be reached without collision.

This part of the state is crucial to ensure the soundness of the search process: before moving to a new state, we should ensure that a path exists to reach it from the previous state. This validation implies finding at least one valid geometric state. Since this process is applied very frequently along the search, it is important to perform it in efficient way. We will show in §4 how we apply a least-commitment strategy.

Geometric Positions: Whenever the symbolic position of a robot or an object changes, a new "geometric position" is created and attached to it. For example, action 1 `(GOTO)` in the plan illustrated by figure 2 entails a new position for robot `F1`. Note that a new geometric position is associated to a symbolic position if it appears again in the symbolic plan (eg. `P_TI_TA_BB_F1` specifies a position for robot `F1` where it can grasp `BB`; it appears in actions 1 and 4 and will perhaps result in two different configurations). The main function of a geometric position is to maintain a geometric instance of the symbolic position, during the plan search process. Potential instances are stored in an "accessibility list" (§4).

There are two types of links between geometric positions that are established as a consequence of the actions that created them:

- **general_motion** links between geometric positions that have the same **roadmap-type** (§2.2). They correspond to robot motion actions (`GOTO`).
- **roadmap_switch** links that correspond to a transition between geometric positions with different **roadmap-type** (§2.2). For instance, `GRASP/UNGRASP` actions cause a robot composition/decomposition and consequently a switch between roadmap types (fig. 2).

We will see below how accessibility lists are handled for the instantiation of symbolic positions.

4 Accessibilities list management

As mentioned earlier, an accessibility list associated to a geometric position, is a set of possible configurations for one robot/object. An element from such list is a roadmap node. The links between geometric positions are translated into links between elements of accessibility lists.

Figure 3 shows the accessibility list that are progressively built during the search for the plan illustrated by figure 2.

Construction of accessibility lists: For example, in Figure 3-1, the accessibility list associated to `P_TI_TA_BB_F1` is composed of 4 elements belonging to the transit roadmap of `F1` and which are all linked to the initial robot configuration.

We will study the mechanisms related to the accessibility lists and to the geometric states through this example (fig. 3).

In the initial state, only one position (roadmap node) is given for each robot/object. Action 1 is a transit action for `F1`. It corresponds to a `general_motion` link and is planned in order to reach any roadmap node that can be linked to a `BB` position. In order for action 1 to be applicable, the planner has to find at least one valid (collision-free) path. Let us assume that the planner has chosen a path that leads to position (b) (fig. 3-1).

Action 2 is the grasp of `BB` by `F1`. A roadmap switch is performed in order to allow motion of the new robot `F1-BB`. In our example, only roadmap node (b) allows to grasp `BB` in its initial position. Therefore the accessibility list will have only one element. Since the previous roadmap nodes of `F1` and `BB` are in a valid geometric state, the composite roadmap node is also valid. No further collision test is necessary to validate the new geometric state.

The third action is similar to the first one (GOTO), but it implies the composite robot `F1-BB` which has to reach a position where it can release `BB`. As shown in figure 3-1, the transfer roadmap allows to place it in many places. Since there are no other constraints (for the moment), a "lazy" choice is performed: the chosen position is the one that implies the least collision checks: `BB` is not moved.

The next actions (fig. 3-2) are similar to the first ones. `F2` moves to carry the Flat-box `FB`. The collision checks are done with the position of `F1` and `BB` as they computed for the last geometric state. The resulting positions are not the best, but

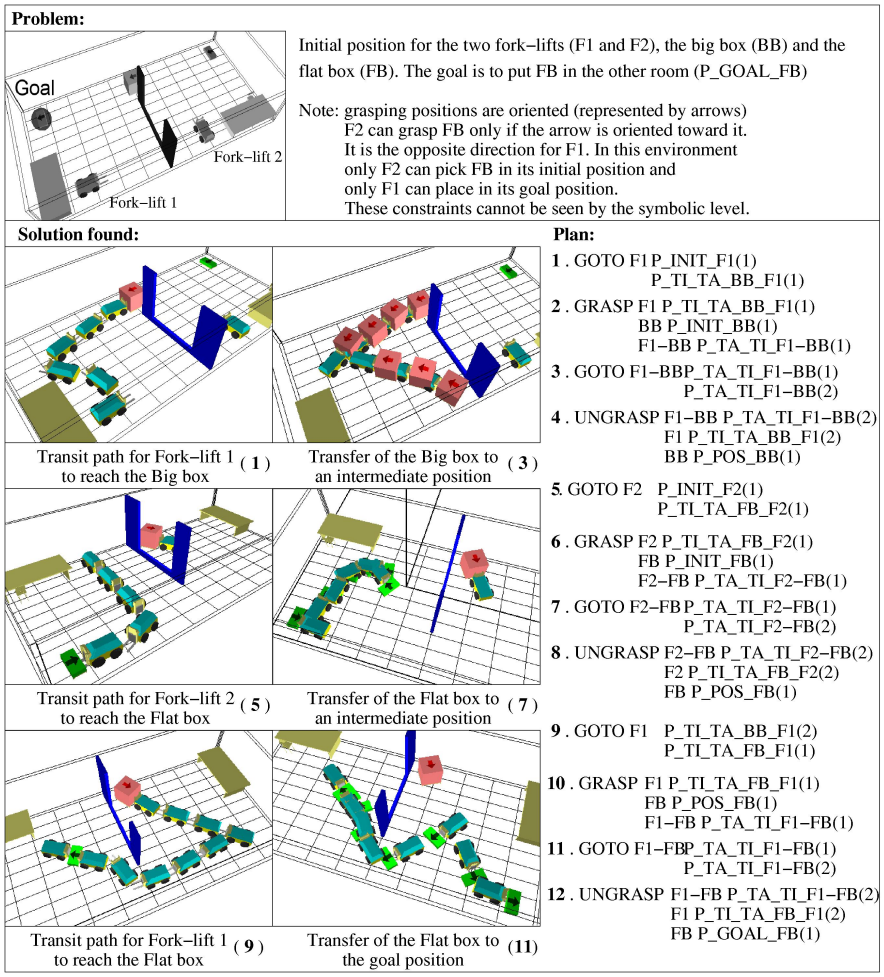

Fig. 2. Example of two fork-lifts (`F1` and `F2`) that have to carry the Flat box `FB` in the other room. The top figure shows the environment and the initial and goal positions. The second part shows the solution plan that has been produced by aSyMov (12 steps). Robots trajectories are illustrated in the left part.

with the current knowledge of the planner they are valid and entail a minimum number of collision checks.

However the next step will not be feasible with the current position instantiations. We describe below how our back-propagation mechanism will allow to find new instances that satisfy the new constraints without changing the actions of the current plan.

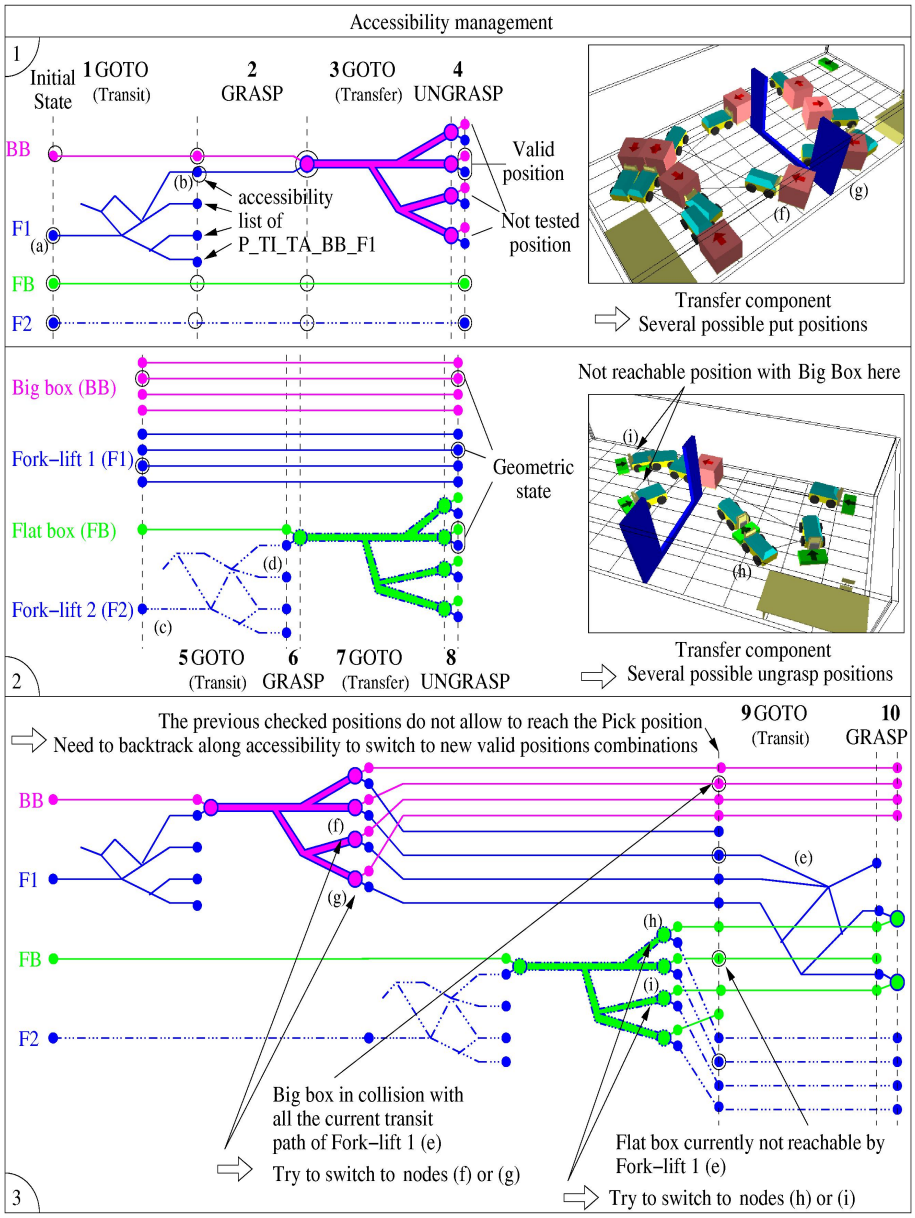

Fig. 3. Illustration of the management of accessibility lists and of position instantiation. (1) and (2) show how the accessibility lists are built respectively for actions 1-4 and 5-8. In (3), geometric instantiation of action 10 involves a back-propagation of the accessibility to validate new geometric states.

Back-propagation: Back-propagation through accessibility lists allows to change the positions of the robots due to geometric constraints without modifying the symbolic level. Figure 3-3 represents the accessibility computed in figures 3-1 and 3-2. Then a new transit action for F1 is applied. When F1 tries to pick up the FB, two accessibility roadmap nodes are found. So it is a possible to achieve this action but not with the previous geometric state. Indeed, it is necessary to find a new valid configuration for FB in action 7.

A state has not only one but a list of geometric states. Since there are several possible valid configurations combinations, there are several geometric states. When a geometric state that has been previously computed is found incompatible with new constraints, a back-propagation mechanism is invoked that restrains the accessibility lists in the previous states. In the example, the only possible roadmap nodes after the transfer of the FB by F2 (action 7) are (h) and (i).

Back-propagation is performed to find if a new valid geometric state is possible for each action that creates a general_motion link. For example, it is possible for F2 with the previous geometric state to carry FB to (h) through the transfer component.

This kind of backtrack is not only used for roadmap nodes connectivity problems but also for collision checking problems. In figure 3-2 it is obvious that with the current position of BB, it is not possible for F1 to reach FB. Even if F2 reaches (h), the validation part of the transit of F1 always fail. This is due to the initial configuration of F1 and to all the robots/objects configurations that collide with it during the path checking process. In other words, the combination of the configurations of F1 and BB is impossible. This impossibility is propagated into the data that store the possible combinations of the accessibility elements. Thus the planner performs a backtrack until action 3 where a new instance of P_TA_TI_F1-BB has to be chosen: (f) or (g) are two positions where F1 can place BB in order to "clear the way".

Thanks to this procedure it is possible to find instantiations that are satisfy all the steps of the solution in figure 2. Now, if all accessibility combinations fail then it is not currently possible to reach the next state. However, further roadmap extension may exhibit a valid path to reach states where the validation process has previously failed.

5 Conclusion and future work

aSyMov combines the capacity of symbolic and geometric planners in a far more elaborated way than simple hierarchy. Both influences are taken into account, but without adding too much constraints on the symbolic part when it is not necessary. A first version of aSyMov is implemented and is able to produce valid plans for intricate environments.

However aSyMov is still an on-going work and we are considering several potential improvements. Two of them will certainly provide substantial performance improvements: (1) better roadmap extension techniques and (2) better heuristics for the search process.

Concerning roadmap extension and exploration there is a need for data structures and algorithms that are better adapted to changes in the environment (e.g. adding or retrieving obstacles from the environment). Besides, the classical tree-like data structure for connected components is not appropriate for multi-robot applications. If only one edge collides with another object, then the whole component is unusable. We already use cyclic graph but need a good method to keep only the best edges.

For the search process guidance the first version of the planner implements a simple heuristic based on symbolic plans with a weighted probabilistic choice. We are investigating how we can introduce information on roadmap connectivity aspects. This will certainly provide valuable heuristic hints.

References

1. 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.
2. R. Alami, R. Chatila, S. Fleury, M. Ghallab and F. Ingrand: "An Architecture for Autonomy". In *The International Journal of Robotics Research*, Volume 17, Number 4, April 1998
3. R. Alami, T. Siméon and J.P. Laumond: "A geometrical Approach to planning Manipulation Tasks. The case of discrete placements and grasps". In *International Symposium on Robotics Research*, Tokyo, August 1989
4. 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.
5. F. Gravot, R. Alami and T. Siméon "Playing with Several Roadmaps to Solve Manipulation Problem". In *IEEE International Conference on Intelligent Robot & System*, Lausanne, September, 2002
6. F. Gravot, S. Cambon, R. Alami, M. Ghallab: "From abstract to concrete planning: integrating motion, manipulation and task planning ". Technical Report LAAS number 03008, January 2003
7. J. Hoffmann and B. Nebel: "The FF Planning System: Fast Plan Generation Through Heuristic Search". In *Journal of Artificial Intelligence Research*, 2001, Vol. 14, pp. 253-302
8. J. Hoffmann "Extending FF to Numerical State Variables". In *Proceedings of the 15th European Conference on Artificial Intelligence*, Lyon, France, July 2002.
9. 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
10. Y. Koga & J.-C. Latombe: "On Multi-Arm Manipulation Planning". In *IEEE International Conference on Robotics and Automation*, vol. 2, pages 945–952, 1994.
11. A. Sahbani, J. Cortès and T. Siméon "A Probabilistic Algorithm for Manipulation Planning under Continuous Grasps and Placement". In *IEEE International Conference on Intelligent Robot & System*, Lausanne, September, 2002
12. T. Siméon, J-P. Laumond, C. Nissoux: "Visibility based probabilistic roadmaps for motion planning". *Advanced Robotics Journal*, Vol. 14, n^o 6, 2000.