A geometrical approach to planning manipulation tasks. The case of discrete placements and grasps

Rachid Alami, Thierry Simeon, Jean-Paul Laumond

A Geometrical Approach to Planning Manipulation Tasks (3). The Case of Discrete Placements and Grasps

Rachid ALAMI, Thierry SIMEON, Jean-Paul LAUMOND
LAAS-CNRS
Groupe Robotique et Intelligence Artificielle
7, Avenue du Colonel Roche
31077 Toulouse cedex (France)

January 20, 1989

Abstract

This paper presents a new geometrical formulation of the manipulation task planning problem in robotics. The problem is shown to be a constrained instance of the coordinated motion planning problem for multiple moving bodies. The constraints are related to the placement and the motion of objects, and can be expressed geometrically.

We give a general paradigm for building Manipulation Task Planners based on the proposed formulation. A manipulation task appears as a path in the configuration space of the robot and all movable objects. A *manipulation path* is a sequence of constrained paths: *transit-paths*, where the robot moves "alone", and *transfer-paths*, where the robot holds an object. The approach consists in building a *Manipulation Graph* that models the connectivity between certain regions in the global configuration space by transit-paths and transfer-paths.

This approach is then applied to the case of a finite number of object placements and grasps. The nodes in the Manipulation Graph correspond to well identified configurations and the edges correspond to paths built from a series of configuration space slices. An implemented system is presented and discussed.

1 Introduction

1.1 Informal problem statement

Let us begin with an informal presentation of the manipulation problem in robotics. We consider an environment containing a robot which has to move and to displace objects without colliding obstacles. Obstacles are stationary bodies, while objects are bodies that can move but only if they are moved by the robot.

There are specific regions in the workspace where the objects can be placed. There are specific geometric relations between the robot and an object to be satisfied in order to allow the object to be grasped. When an object is grasped, it "follows" the robot until an ungrasp action is performed.

In such a context, a manipulation task planning problem consists in finding a sequence elementary robot motions and grasping and ungrasping actions that will allow to reach a given state starting from an initial state. Section 2 gives an example of such a problem.

1.2 Informal approach to the problem

This paper presents a geometrical formulation of the problem. Indeed, a manipulation task can be characterized by a geometrical description of the environment, at any instant of the manipulation. Our approach is similar to those developed for the robot motion planning problem. It consists in transforming the problem of planning a manipulation task into the problem of finding a path in a certain space.

Let us consider the configuration space of all movable bodies (robot and objects). A manipulation task appears as a particular path in this space.

The various constraints due to the manipulation context can be geometrically expressed. For example, the constraints on object placements invalidate some configurations; consequently, the search space is only a subset of the collision-free configuration space. Furthermore the paths which model the elementary actions of a manipulation task are paths which are not included in the "full" space, but in some n-dimensional spaces (where n is the number of degrees of freedom of the robot). These elementary paths are called *transit-paths* and *transfer-paths.*

Section 4 details the geometric model of the manipulation problem, and shows that it is a very constrained instance of the coordinated motion planning problem. Various geometrical structures are introduced in order to decompose a certain configuration subspace into components whose adjacency relation characterizes the solution space of the manipulation problem. We then provide a paradigm for building a *Manipulation Graph* whose connected components

give the existence of a solution to a given problem.

This paradigm is applied (section 5) to the case of a manipulation environment where the number of object placements and grasps is finite. We describe an implementation of manipulation task planner and illustrate its use in the case of a robot in translation amidst polygonal objects and obstacles in a 2-d environment. Finally we discuss several issues induced by this approach and that we intend to explore in order to advance in a better understanding of manipulation.

2 An example

We have implemented a system that is to able solve problems similar to the manipulation task planning problem illustrated in figure 1. Let us analyze quickly such an example. We have a robot and two objects A and B.

The only difference between the initial and the final state is B position. However, several difficulties must be tackled:

- the robot must find out that it has to displace A because it forbids access to B;
- it has to choose how to grasp A and where to put it (sequence 1-2);
- the only solution is to put it on the floor, to re-grasp it and to put it again in its initial position (sequence 3-4) in order to displace B before re-displacing again A (sequence 5-6-7)...

Several observations could be made:

- the example is voluntarily very constrained in order to show different aspects of the problem;
- it should be nice to specify only partially the target state;
- a simple robot motion request could be transformed into a complex manipulation problem if some object obstructs the path;
- a slight change in the bodies size or shape or in the robot structure may entail a completely different plan for achieving the "same" task;
- an interesting decision problem could be added when an object does not obstruct completely a path; nevertheless, its displacement may simplify the task execution. Various criterias may be taken into account as the number of Pick-and-Place steps or the total robot trajectory length.

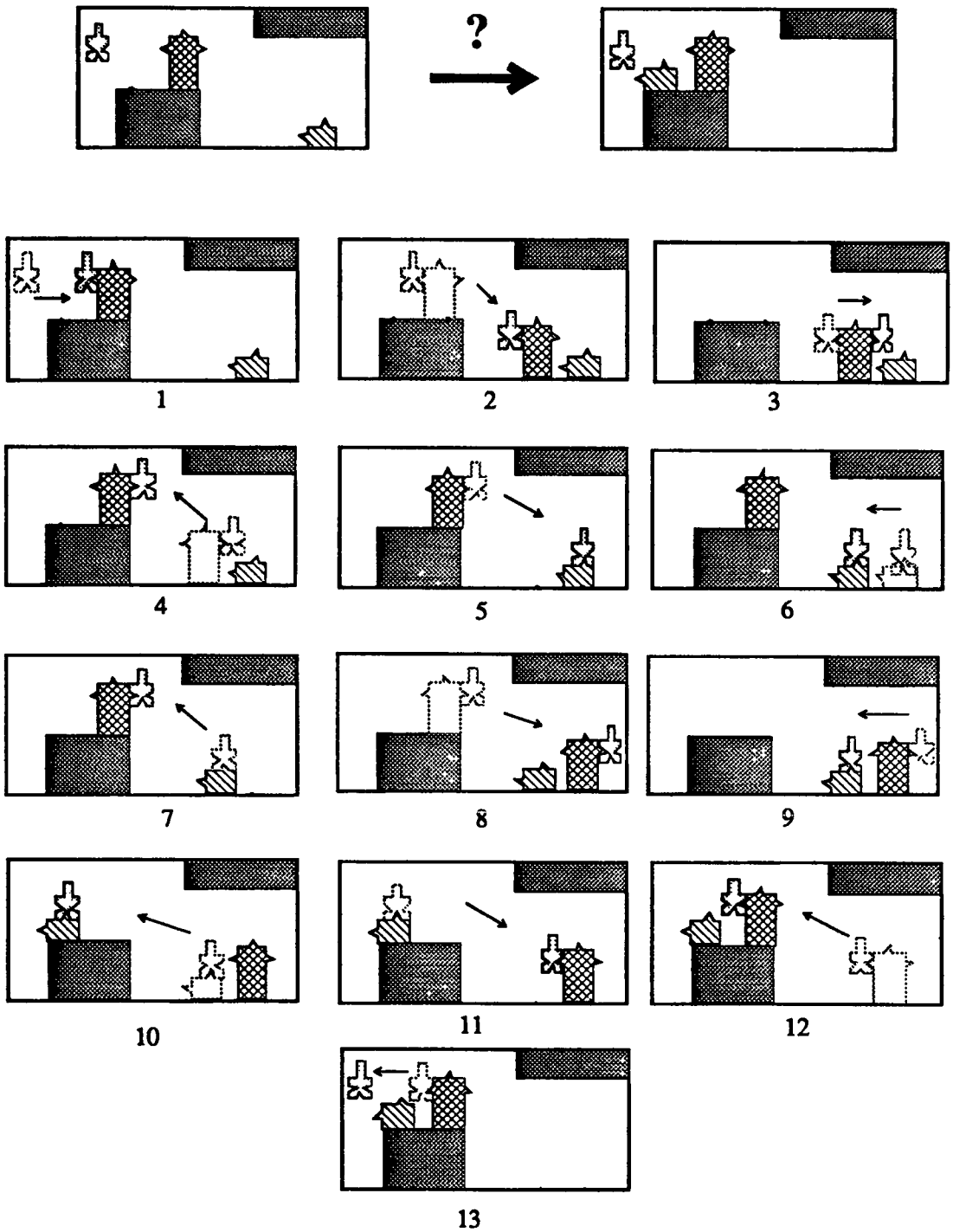


Figure 1: A manipulation task

3 Related topics

We are not aware of a similar research work. However, the problem we tackle is closely linked to three active research fields: Coordinated Motion Planning Problems, Automatic Task Level Robot Programming, General Action Planners. Let us discuss briefly how they interact with the problem at hand.

3.1 Coordinated Motion Planning

Several authors have explored the motion planning problem for multiple independent moving bodies [12] [2]. [17] shows that the problem is NP-hard for the simple case of several disks amidst polygonal obstacles.

The manipulation task planning problem appears clearly to be a very constrained instance of the coordinated motion planning problem, but the existing algorithms do not apply directly. However, we may take advantage of the configuration space structuring techniques available for several instances of the coordinated motion problem and extend them in order to take into account the specific constraints introduced by the manipulation problem [5].

3.2 Automatic Task Level Robot Programming

Several systems [7] [4] [8] tackle the problem of synthesizing a sequence of effector level actions starting from a task level description of a Pick-and-Place task. Here again, authors rely heavily on geometric reasoning and more particularly on motion planning. In order to cope with the problem complexity, all proposed systems rely on a decomposition of a Pick-and-Place operation into several steps: gross motion, fine motion, grasping... One major difficulty lies in the interdependency between these steps. Indeed, in the general case, a Pick-and-Place operation is not easily decomposable. The manipulation task planning problem adds an other interdependency between successive Pick-and-Place operations.

No existing system tackles the manipulation problem as it has been defined above. However, there are systems that solve a particular instance of the general problem. For example, HANDEY [14] is able to plan a re-grasping operation using a table; a simple Pick-and-Place problem is then solved by performing several Pick-and-Place operations.

3.3 General Action Planners

The manipulation problem could be in some way assimilated to the classical "block world" planning problem [3] [10]. However, these methods are not adapted to take into account

the full complexity of the problem. Indeed, as we mentioned above, the knowledge that is necessary in order to really solve the problem - by producing elementary collision-free robot motions and gripper actions - relies heavily on motion planning. The task decomposition is closely linked to the robot structure and to the shape of objects and obstacles.

But still we think that a general action planner system could be used if one is able to exhibit relevant heuristics that will allow to control a Manipulation Task Planner - such as the one described in section 5 - by defining intermediate goal states for instance.

4 A geometric model of the manipulation problem

This section establishes a geometrical framework for the manipulation problem.

4.1 Manipulation task and Configuration Space(s)

The environment is a 3-dimensional euclidean space; its contains three kinds of bodies:

- obstacles, i.e. fixed bodies
- objects, i.e. movable bodies
- a robot

For the robot and for each object we consider its associated configuration space. Object configuration spaces are 6-dimensional; the robot configuration space of a robot is n-dimensional, where n is its number of degrees of freedom. Each configuration space has the topology induced by the Hausdorff distance between bodies in the euclidean space [16]. Let **CS** denote the cartesian product of all object and robot configuration spaces.

In the following, we will say that c is an incompletely specified configuration when some parameters in c are left unspecified; besides, we will say that a configuration c' "verifies" c when it is included in the subspace defined by c. Such a terminology will be used in order to denote partially specified goals.

Furthermore, we introduce a function $FREE$ which gives, for each domain of **CS**, the set of its free configurations (i.e. configurations where the bodies do not overlap).

A manipulation task is clearly a particular path in $FREE(\mathbf{CS})$. Alas ! the converse does not hold: all paths in $FREE(\mathbf{CS})$ do not necessarily correspond to a manipulation task. Indeed a manipulation path is a very constrained path in $FREE(\mathbf{CS})$. We have now to define geometrically these constraints. There are two types of constraints:

- constraints on the placements of objects; these constraints model the physics of the manipulation context (any object must be in a stable position in the environment),
- constraints on object motions; any object motion is a motion induced by a robot motion.

Each object or robot has a fixed reference frame that determines its position and orientation in the euclidean space. If c is a configuration (of an object or of the robot), $E(c)$ designates the position and the orientation of its reference frame relatively to the environment reference frame. Note that E is bijective for the objects, but it is not for a robot in general (in the case of a manipulator, E corresponds to its direct geometrical model [9]).

4.2 Placement constraints

All configurations in $FREE(\mathbf{CS})$ do not necessarily correspond to a physically valid environment configuration. For example an object can not "levitate", and must be in a stable position. Geometrically speaking, we have to reduce the space of free configurations to a subspace which contains all valid configurations. These constraints concern only the objects. For example, a polyhedron could be constrained to be placed only on a horizontal face of an obstacle or of an object (which is in a stable position); the placement constraints will then define a finite number of 3-dimensional manifolds in the configuration space of the object.

We call **PLACEMENT** the subspace of $FREE(\mathbf{CS})$ containing all valid placements for all objects, i.e. placements which respect the physical constraints of the manipulation context. With this definition, all the objects have a fixed and known geometrical relations with the obstacles or with other objects.

PLACEMENT is not more precisely defined; the definition depends on the context, and appears clearly for each context. For a mobile robot in a 2-dimensional euclidean space, amidst movable objects, **PLACEMENT** $= FREE(\mathbf{CS})$. Work has been (and must be) developed in order to compute the stable placements of an object in a 3-dimensional environment. In section 5, we will suppose that each object has a finite number of placements in the environment; then **PLACEMENT** will appear as a finite union of n-dimensional manifolds, where n is the number of degrees of freedom of the robot.

Remark: It would be interesting to take into account these constraints by an alternative approach that would consider the gravity as a particular "robot" in the environment, acting the objects when they are placed in a unstable position. But this is another story.

4.3 Motion constraints

Let us consider a homogeneous transformation T in the euclidean space. We define a grasp mapping G_O^T as a mapping from the configuration space of the robot into the configuration

space of a given object O, that verifies $E(G_O^T(c)) = T(E(c))$. This mapping models the geometrical relation which is defined by a grasping operation. Such mappings define geometrically the semantics of grasping for a particular manipulation context. They can be in finite or infinite number, and can be given explicitly (as in section 5) or implicitly (they are defined for example by a contact relation between the robot or the object as in [5]).

Definition A *transfer-path* is a path in $FREE(\mathbf{CS})$ such that there is one object O and one grasp mapping G_O^T verifying:

- the configuration parameters of any object $O' \neq O$ are constant along the path
- for any configuration of the path, $G_O^T(r) = o$, where r and o designate respectively the configuration parameters of the robot and O.

Two configurations of $FREE(\mathbf{CS})$ connected by a transfer-path are *g-connected.*

We call **GRASP** the subspace of $FREE(\mathbf{CS})$ containing the configurations which are *g-connected* with a configuration of **PLACEMENT**. Let us denote **VALID** = **PLACEMENT** ∪ **GRASP**. The relations between the various subspaces of **CS** are shown figure 2.

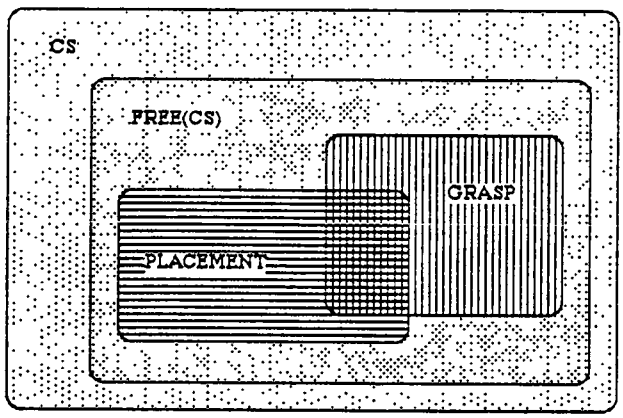


Figure 2: The various subspaces of **CS**

Definition: A *transit-path* is a path in $FREE(\mathbf{CS})$ such that the configuration parameters of the objects are constant along the path. Two configurations in $FREE(\mathbf{CS})$ connected by a transit-path are *t-connected.*

Remark: a transit-path is included in **PLACEMENT** (but every path in **PLACEMENT** is not necessary a transit-path).

We are now in position for defining any manipulation task as a manipulation path in $FREE(\mathbf{CS})$:

Definition: A *manipulation-path* is a path in $FREE(\mathbf{CS})$ which is a finite sequence of transit-paths and transfer-paths. Two configurations in $FREE(\mathbf{CS})$ connected by a manipulation-path are *m-connected.*

A manipulation problem can then be defined as:

> **Manipulation problem:** An initial configuration i and a final (completely or incompletely specified) configuration f being given, does there exist a configuration verifying f which is *m-connected* with i ? If the answer is yes, give a *manipulation path* between i and some configuration verifying f.

4.4 A very simple example

Let us discuss a very simple example in order to illustrate the formulation we propose. We consider a locomotive engine and a wagon moving on a track linking two hills (figure 3). There is no obstacle; numbers on the track represent curvilinear abscissa values. The reference point of each body is the center point of the body. Figure 3-5 shows $FREE(\mathbf{CS})$; it has two connected components according to the fact that the engine is at the left of the wagon or not. Figure 3-6 shows the two segments modeling **GRASP**. The stable positions of the wagon in the environment are shown in bold line in figure 3-2 (the wagon can not stand on the slopes without being "grasped" by the engine). **PLACEMENT** consists of six connected components (figure 3-7).

Motions constraints appear clearly: transit-paths are horizontal segments in **PLACEMENT**; transfer-paths are segments included in **GRASP**.

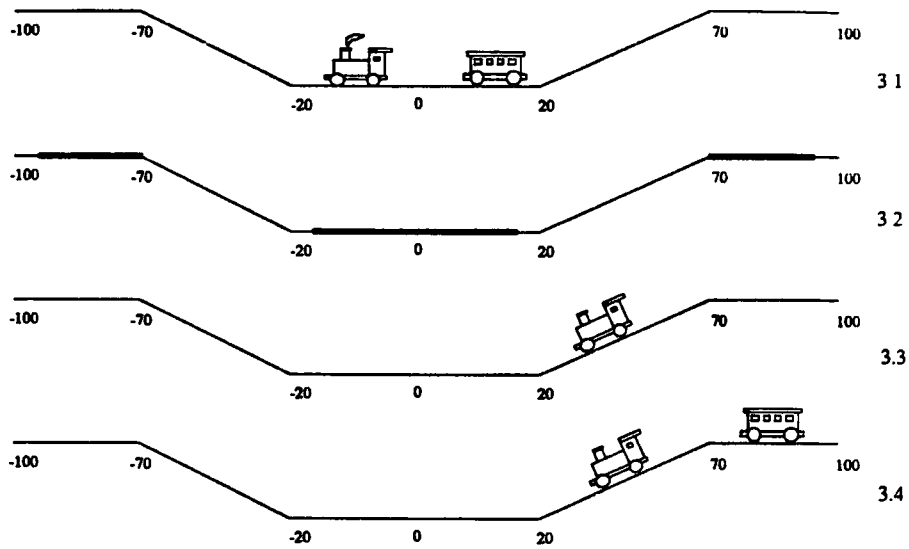
Let us consider an initial configuration c (figure 3-1) and a final uncompletely specified configuration f: the engine does not "grasp" the wagon, and its position is given (figure 3-3). f is the subspace consisting of three vertical segments (figure 3-8). Figure 3-8 shows an example of a solution. The final configuration c' is illustrated in figure 3-4.

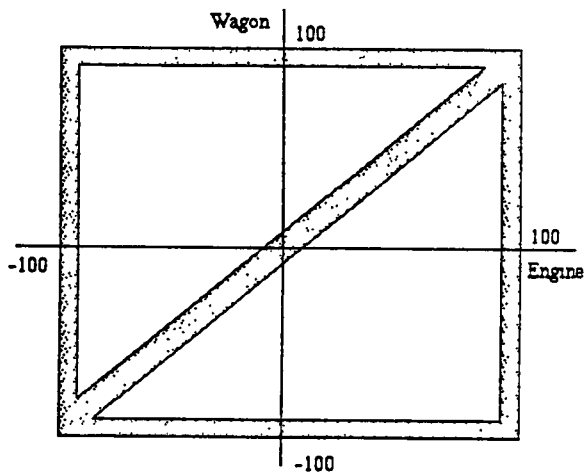
4.5 The Manipulation Graph

Our various definitions lead to a fundamental property which models the structure of the solution space:

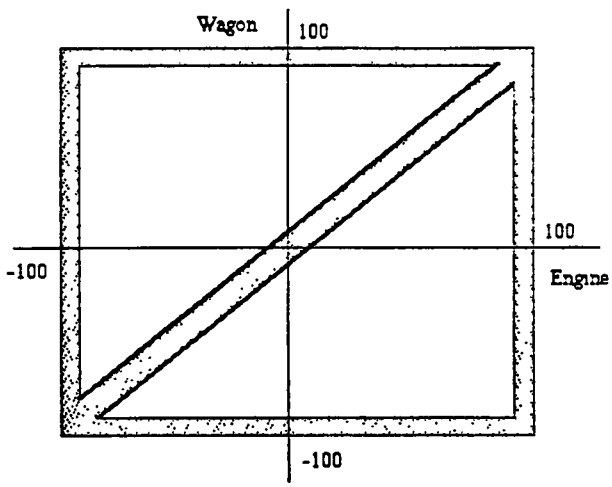
Property: A transit-path and a transfer-path are connected iff both have an extremity in common in **PLACEMENT** $\cap$ **GRASP**.

We prove in [6] that two configurations which are in a same connected component of **GRASP**$\cap$ **PLACEMENT** are *m-connected.* This property leads to reduce the problem. Indeed, it suffices to study the connectivity of the various connected components of **GRASP** $\cap$

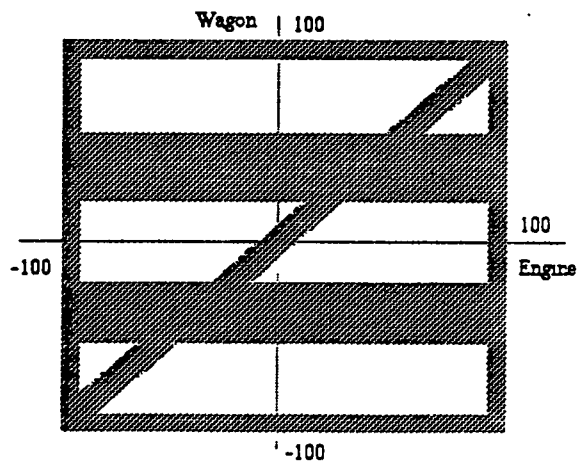




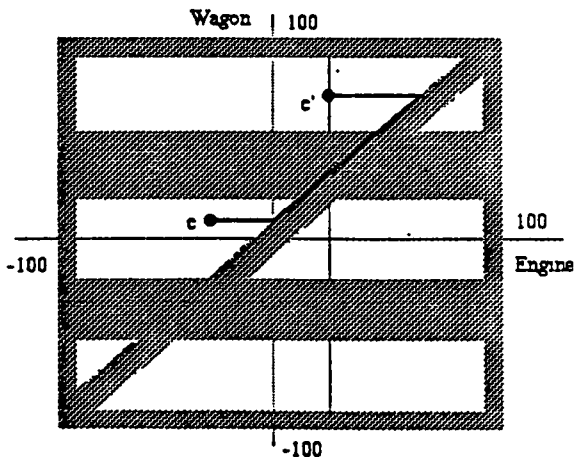
3.5 : FREE(CS)



3.6 : GRASP



3.7 : PLACEMENT



3.8 : a solution

Figure 3: A very simple example

PLACEMENT by transit-paths and transfer-paths. This study leads to define a graph whose nodes are the connected components of **GRASP** $\cap$ **PLACEMENT**. There are two types of edges. A *transit* (resp. *transfer*) edge between two nodes indicates that there exist a transit-path (resp. transfer-path) path linking two configurations of the associated connected components.

This graph is called a *Manipulation Graph (MG)*. It verifies the fundamental property:

Property: An initial configuration i and a goal (completely or uncompletely specified) configuration g being given, there exists a configuration f verifying g and m-connected with i iff:

- there exist a node N_i in MG and a configuration c_i in the associated connected component of **GRASP** $\cap$ **PLACEMENT**, such that i and c_i are t-connected or g-connected;
- there exist a node N_f in MG and a configuration c_f in the associated connected component of **GRASP** $\cap$ **PLACEMENT** such that:
  - c_f and f are t-connected or g-connected;
  - N_i and N_f are in same connected component of MG

This property shows that the problem is decidable. We give in [6] a detailed proof in the general case. It uses a particular cylindrical algebraic decomposition of **CS** (as in [11] for the piano mover problem).

In order to use this method for particular instances of the problem, one needs:

- to compute the connected components of **GRASP** $\cap$ **PLACEMENT** ,
- to determine the connectivity of these connected components using transit-paths and transfer-paths,
- and to provide a method for planning a path in a given connected component of **GRASP** $\cap$ **PLACEMENT**

Some results have already been obtained.

The work reported by Wilfong in [15] is included in the framework we propose: it considers the 2-dimensional case, where only translations are allowed and suppose (with our notation) that **VALID** $= FREE(\mathbf{CS})$; it shows that the problem is NP-hard (resp. PSPACE-hard) when the final configuration is incompletely (resp. completely) specified; it gives an $O(n^3 \log^2 n)$ algorithm for polygons in translation, the number of grasp mappings being finite.

We have solved in $O(n^4)$ the associated decision problem for a circular robot and a circular object amidst polygonal obstacles [5]; in this case, **VALID** $= FREE(\mathbf{CS})$ and the number

of grasp mappings is infinite; they are defined from any configuration where the robot and the object are in contact. The path planning method and the study of adjacency relations is done by a cell decomposition of $\mathbf{GRASP} \cap \mathbf{PLACEMENT}$ (which, in this case, is equal to the 3-dimensional contact space); this decomposition is obtained by a "projection" of the cell decomposition of the 4-dimensional free configuration space of two disks presented in [12].

5 A solution for the case of discrete placements and grasps

In this section, we present a description of a Manipulation Task Planner for the case of discrete placements and grasps for objects. It is directly derived from the general scheme discussed in section 4. It is based on the fact that the connected components of $\mathbf{GRASP} \cap \mathbf{PLACEMENT}$ are given *a priori* by the discretization and that the construction of transit-paths and transfer-paths can be obtained using a collision-free path planner for a robot amidst stationary obstacles.

It leads to an effective construction of the Manipulation Graph.

For simplicity reasons, we give a formulation considering only two objects. The extension to a finite number of objects is straightforward. The presentation will be illustrated using the example of figure 1, i.e. a 2-d world where all bodies are polygonal and are allowed to move only in translation. However, the solution we propose is general.

5.1 Notations

We designate the robot by R and the objects by A and B. Let r, a and b be the configuration parameter vectors and $\mathbf{CR}$, $\mathbf{CA}$ and $\mathbf{CB}$ the associated configuration spaces. Let n be the dimension of $\mathbf{CR}$. The configuration space of all movable bodies is: $\mathbf{CS} = \mathbf{CR} \times \mathbf{CA} \times \mathbf{CB}$.

Object placements: We assume that each object has a finite number of possible placements in the environment that will physically correspond to stable positions when the robot does not hold the object.

We designate by $p_A^1, \ldots p_A^i \ldots \in \mathbf{CA}$ and by $p_B^1, \ldots p_B^j \ldots \in \mathbf{CB}$ the authorized placements for A and B respectively (figure 4).

Remark: we may also give explicitly - or give means to compute - all authorized placements combinations $(p_A^i, p_B^i) \in \mathbf{CA} \times \mathbf{CB}$, in order to take into account, for example, the possibility of stacking an object on another object.

Object grasps: We assume that each object has a finite number of possible grasps. A given

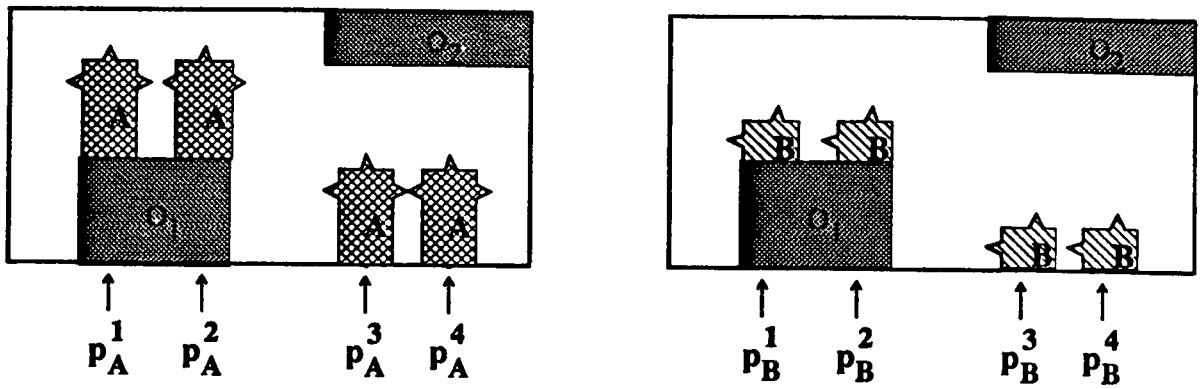


Figure 4: Placements for objects A and B

grasp for object A is specified by providing a mapping G_A^i : **CR** $\longrightarrow$ **CA**.

Let G_A^1, G_A^2 G_A^3 and G_B^1, G_B^2 be the authorized grasps for A and B(figure 5).

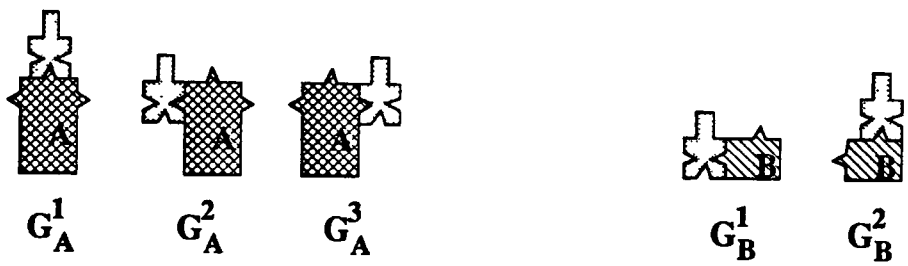


Figure 5: Grasps for objects A and B

5.2 Building the Manipulation Graph

Following section 4, the construction of MG has to be performed by obtaining a decomposition of **GRASP** $\cap$ **PLACEMENT** into connected regions, and by exploring connections between them by transfer-paths and transit-paths.

5.2.1 Building the nodes

Intuitively, **GRASP** $\cap$ **PLACEMENT** is simply the set of all configurations where the robot is authorized to grasp or ungrasp an object in a given placement, taken into account all possible placement combinations for the other objets.

Let $\mathbf{I}(grasp\ ;\ placementA\ ;\ placementB)$ designate the set of all free configurations where the robot can grasp one object using *grasp* while objects are placed in *placementA* and *placementB*.

Thus, $\mathbf{GRASP} \cap \mathbf{PLACEMENT}$ is the union of all $\mathbf{I}(G_A^i\ ;\ p_A^j\ ;\ p_B^k)$ and $\mathbf{I}(G_B^i\ ;\ p_A^j\ ;\ p_B^k)$.

By definition, we have:

$$\mathbf{I}(G_A^i\ ;\ p_A^j\ ;\ p_B^k) = FREE(\{r \in \mathbf{CR} \mid G_A^i(r) = p_A^j\} \times \{p_A^j\} \times \{p_B^k\})$$

The computation of $\{r \in \mathbf{CR} \mid G_A^i(r) = p_A^i\}$ is done using the robot inverse geometric model. If we assume that the robot is not redundant and that the solution does not include a robot singular configuration, then $\mathbf{I}(G_A^i\ ;\ p_A^j\ ;\ p_B^k)$ consists of a finite (and small) number of configurations.

Each node of *MG* corresponds to a configuration that can be computed easily; it represents a connected component of $\mathbf{GRASP} \cap \mathbf{PLACEMENT}$ reduced to a single configuration.

5.2.2 Building the edges

The second step in building *MG* consists of establishing edges between nodes. Following 4.5, two nodes N_1 and N_2 are linked by a transit (resp. transfer) edge if there exits a transit-path (resp. transfer-path) between two configurations of their associated connected components.

In the case of discrete grasps and placements, all paths involving a node have an extremity in common: the single configuration represented by the node. For a node in $\mathbf{I}(G_A^i\ ;\ p_A^j\ ;\ p_B^k)$, all transit-paths will correspond to paths where the robot moves alone while A and B are in p_A^j and in p_B^k, and all transfer-paths will correspond to paths where the robot holds A using grasp G_A^i while B is in p_B^k. This is why we define the concept of *task state* that denotes the fact that the robot is in a situation where it moves alone, or it is holding a given object.

Task states and CS slices: We define a *task state* by giving for each object its current placement and, for at most one object, its current grasp: $(grasp\ ;\ placementA\ ;\ placementB)$.

When no object is grasped, *grasp* will be noted "_", for example $(\ \_\ ;\ p_A^1\ ;\ p_B^2)$. We call such a state a *transit state*.

When object X is held by the robot, *placementX* will be noted "_", for example $(G_A^2\ ;\ \_\ ;\ p_B^4)$. We call such a state an *transfer state*.

Let $\mathbf{C}(grasp\ ;\ placementA\ ;\ placementB)$ denote the set of configurations in **CS** associated to a given task state. $\mathbf{C}(\ \_\ ;\ p_A^i\ ;\ p_B^j)$ and $\mathbf{C}(G_A^i\ ;\ \_\ ;\ p_B^j)$ correspond to n-dimensional "slices" of **CS** where n is the dimension of **CR**.

For a transit state: $\mathbf{C}(\ \_\ ;\ p_A^i\ ;\ p_B^j) = FREE(\mathbf{CR} \times \{p_A^i\} \times \{p_B^j\})$

i.e. all configurations such that the robot does not overlap object A in placement p_A^i nor object B in placement p_B^j nor the obstacles.

For a transfer state: $\mathbf{C}(G_A^i\ ;\ \_\ ;\ p_B^j) = FREE(\mathbf{CR} \times G_A^i(\mathbf{CR}) \times \{p_B^j\})$

i.e. all configurations such that the robot holding object A in grasp G_A^i does not overlap object B in placement p_B^j nor the obstacles.

Remark: When $\mathbf{C}(G_A^i\ ;\ \_\ ;\ p_B^j) = \emptyset$ the corresponding state is invalid.

A node models a transition between a transit state and a transfer state. We say that the node "belongs" to these states. For example, a node in $\mathbf{I}(G_A^i\ ;\ p_A^j\ ;\ p_B^k)$ "belongs" to the transit state $(\ \_\ ;\ p_A^j\ ;\ p_B^k)$ and to the transfer state $(G_A^i\ ;\ \_\ ;\ p_B^k)$.

Transit edges: A transit edge can be built between a node N and any node N' that belongs to the same transit state and that is "directly" reachable. This simply means that N and N' represent configurations that are in a same connected component of $\mathbf{C}(\ \_\ ;\ p_A^i\ ;\ p_B^j)$.

Transfer edges: A transfer edge can be built between a node N and any node N' that belongs to the same transfer state and that is "directly" reachable. This simply means that N and N' represent configurations that are in a same connected component of $\mathbf{C}(G_A^i\ ;\ \_\ ;\ p_B^j)$ or $\mathbf{C}(G_B^i\ ;\ p_A^j\ ;\_\ )$

We have then to construct two **CS** slices for any given node. However, a given **CS** will be used for a great number of nodes.

Figure 6 represents several nodes in the Manipulation Graph that corresponds to the example. The drawing at the center of the figure represents the node $\mathbf{I}(G_B^2\ ;\ p_A^3\ ;\ p_B^2)$. Transit and transfer edges are built using $\mathbf{C}(\_\ ;\ p_A^3\ ;\ p_B^2)$ and $\mathbf{C}(G_B^1\ ;\ \_\ ;\ p_B^2)$.

Figure 7 illustrates the "links" between several configuration space slices that are traversed by the system when it executes the sequence represented in figure 1. Several states are represented; for each state, the regions in white represent the projection of the connected components of its **CS** slice onto the robot configuration space. In the initial state of figure 1, the robot is in the "left" connected component of $\mathbf{C}(\ \_\ ;\ p_A^2\ ;\ p_B^4)$. The only possible transition (arc 11) is to move the robot until it is able to grasp object A in G_A^2. The transitions sequence is 11-10-7-9-6-5...Note that the solution involves state $(\ \_\ ;\ p_A^2\ ;\ p_B^4)$ twice, but it traverses only once a given connected component of $\mathbf{C}(\ \_\ ;\ p_A^2\ ;\ p_B^4)$.

Remark: In the case of discrete grasps and placements, the manipulation problem can be tackled "directly" using the concept of task state as it has been defined. We have used it before as a high level of abstraction model in order to program and control a Assembly Workcell [1]. However, we have chosen, here, to show how it derives from the general solution scheme presented in section 4.

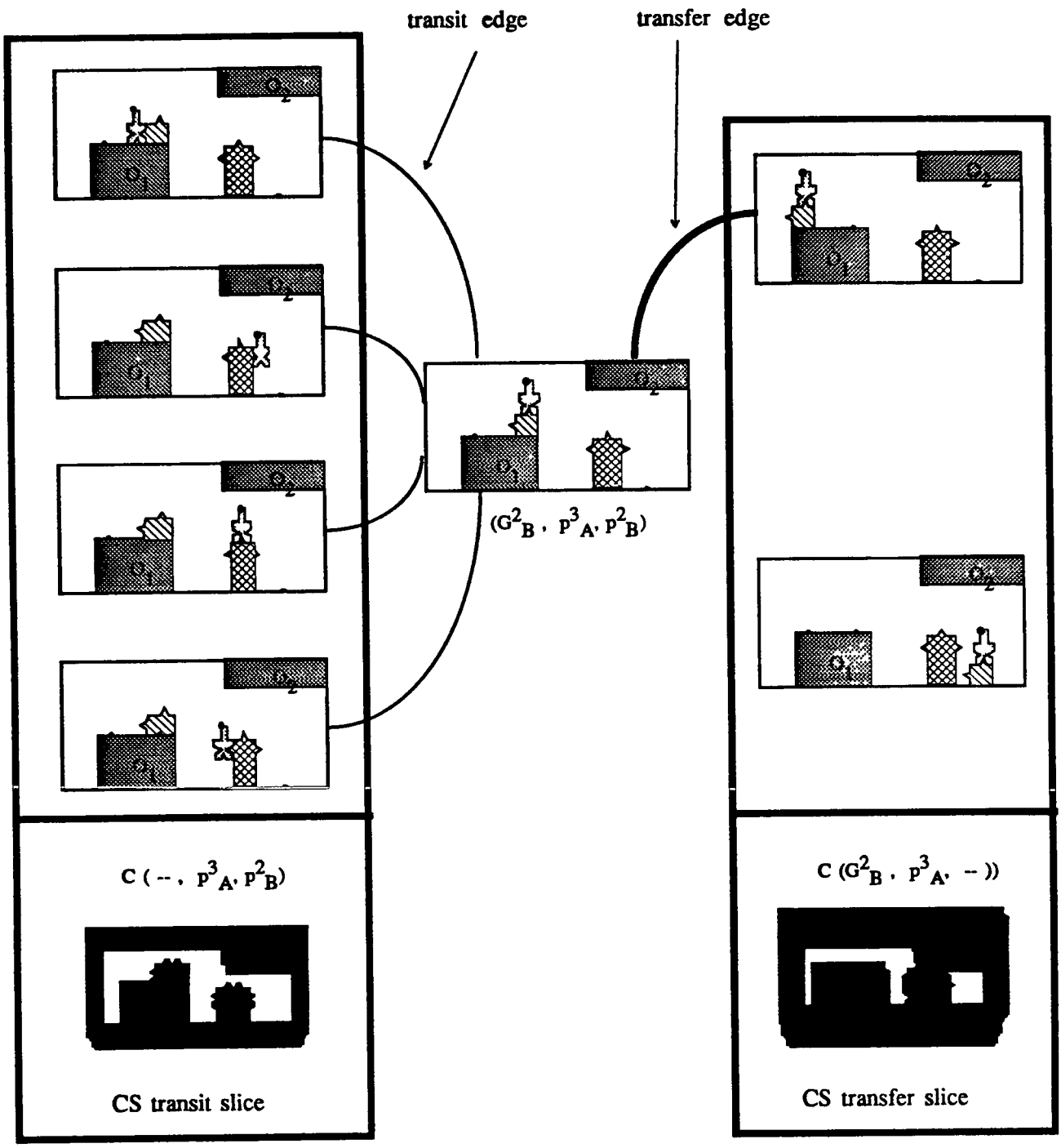


Figure 6: A partial representation of a Manipulation Graph

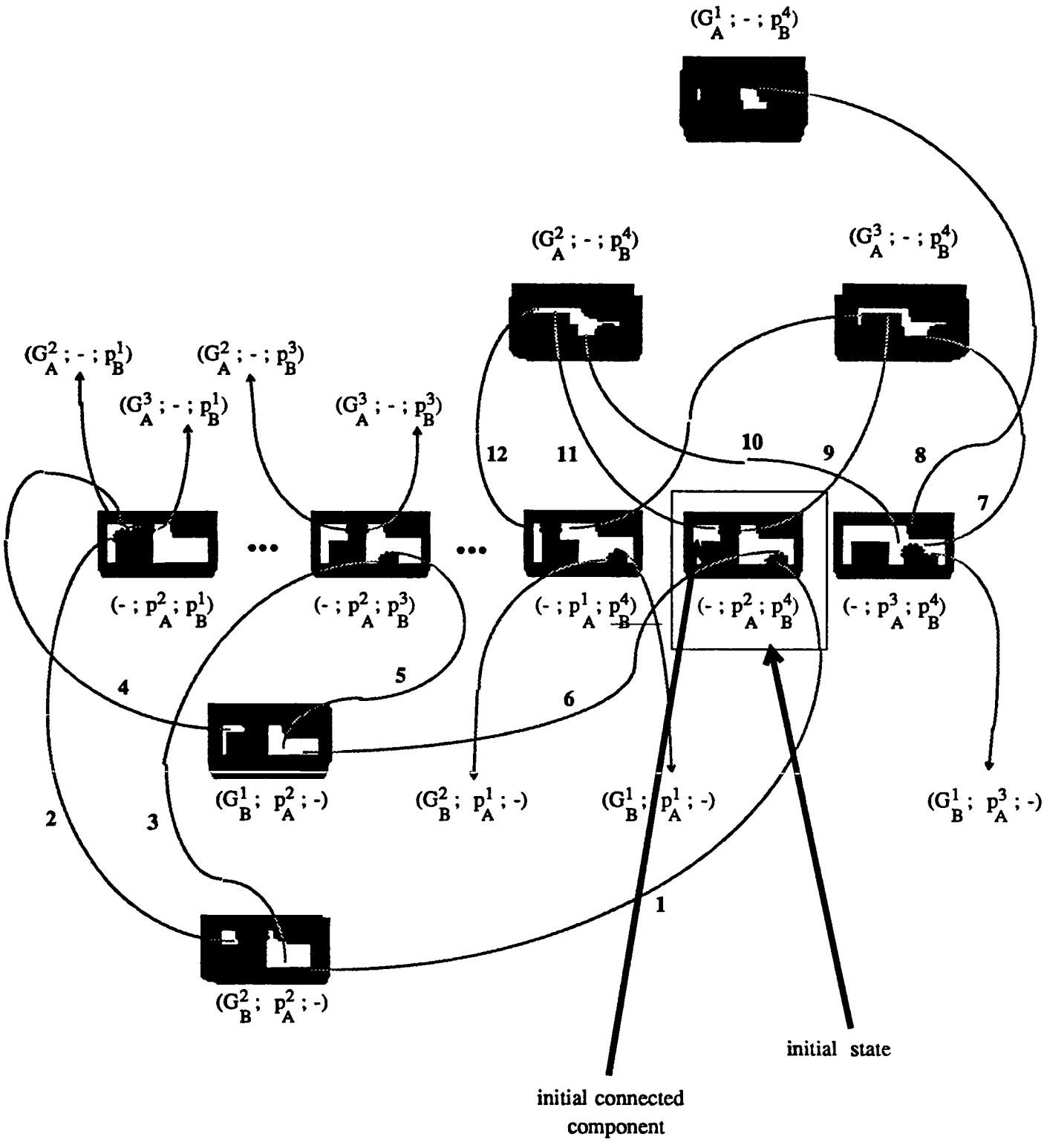


Figure 7: Links between Configuration Space slices

5.3 Search Strategies in the Manipulation Graph

The size of the graph grows rapidly depending on the number of grasps and placements. The number of nodes corresponds to *number of grasps* $\times$ *number of placements* (in our simple example, there are $(3+2) \times (4 \times 4) = 80$ nodes). The number of transit slices is equal to the number of legal placements ($(4 \times 4) - 4 = 12$ in the example). The number of transfer slices is equal to the number of combinations of grasps for an object and placements for the other objects ($3 \times 4 + 2 \times 4 = 20$ in the example).

The cost of building an edge is expensive and depends mainly on the cost of computing a n-dimensional CS slice (where n is the number of degrees of freedom of the robot). However, a CS slice is used several times; for example $\mathbf{C}(\_ ; p_A^j ; p_B^k)$ will be used for all nodes in $\mathbf{I}(G_A^i ; p_A^j ; p_B^k)$ and in $\mathbf{I}(G_B^i ; p_A^j ; p_B^k)$. The first time, it has to be computed; and then, it will only be used in order to find a path.

MG has not to be built completely before execution. It can be explored and built incrementally. Powerful heuristics remain to be explored in order to "drive" the system towards the goal. However, even simple heuristics based only on the distance between the positions of objects allow to limit substantially the construction of the graph.

Note that, if several objects have the same shape and the same grasps and placements, the number of different CS slices to build can be considerably reduced. In the case, similar to the example, of two identical objects with 3 different grasps and 4 placements, we have only 12 transfer slices and 6 transit slices. Then, another way to limit the complexity, when exploring the graph edges, is to consider only a gross approximation of objects shape (by classifying them into a limited number of classes: small, elongated, big...) in order to use a same CS slice for a great number of nodes.

5.4 Implementation and Discussion

5.4.1 Implementation

We have implemented a system based on the method described above. It is composed of two modules: a *Manipulation Task Planner* and a *Motion Planner*.

The *Manipulation Task Planner* builds incrementally the Manipulation Graph and searches solution paths in it. It makes use of the Motion Planner in order to build the different CS slices corresponding to transfer and transit states, to structure them into connected components and to find paths between two robot configurations.

The search in the Manipulation Graph is performed using a A* algorithm. The only heuristic currently used is based on the length of the movable bodies trajectory. The incremental graph construction allows a nice feature: for the first plans, the system is quite slow but it becomes

more efficient progressively as it re-uses parts of the graph already developed.

The *Manipulation Task Planner* is implemented in order to be used for an arbitrary number of objects and does not depend on a specific motion planner module.

In the current implementation we have used a very simple Motion Planner for a polygonal body in translation amidst polygonal obstacles (the obstacles are grown using Minkowsky sum and the trajectory is built using a visibility graph). This has been done mainly in order to demonstrate the feasibilty of the approach and also because such a motion planner produces an exact representation of the free space. Figure 8 shows the plan produced for the example.

```
[1] - (move-to (-9.0 8.0) :via (-16.0 3.5) (-10.0 7.5))
[2] - (grasp :object A :grasp 2)
[3] - (move-to (3.0 -0.5) :via (-5.0 8.0))
[4] - (ungrasp :object A)
[5] - (move-to (10.0 -0.5) :via (2.0 0.0) (2.0 0.5) (3.0 2.0) (5.0 3.0) (8.0 3.0) (10.0 2.0) (11.0 0.5) (11.0 0.0))
[6] - (grasp :object A :grasp 3)
[7] - (move-to (-2.0 8.0) :via (5.0 4.5) (2.0 8.0))
[8] - (ungrasp :object A)
[9] - (move-to (9.0 -4.0) :via (-1.0 7.5) (8.0 -3.5))
[10] - (grasp :object B :grasp 2)
[11] - (move-to (3.0 -4.0))
[12] - (ungrasp :object B)
[13] - (move-to (-2.0 8.0) :via (2.0 -3.5) (-1.0 7.5))
[14] - (grasp :object A :grasp 3)
[15] - (move-to (16.0 -0.5) :via (2.0 8.0) (5.0 4.5) (12.0 4.0) (14.0 3.0))
[16] - (ungrasp :object A)
[17] - (move-to (6.5 -1.5) :via (17.0 0.0) (17.0 0.5) (16.0 2.0) (14.0 3.0) (11.0 3.0) (9.0 2.0) (7.0 -0.5))
[18] - (grasp :object B :grasp 1)
[19] - (move-to (-12.5 7.0) :via (-1.5 7.0))
[20] - (ungrasp :object B)
[21] - (move-to (9.0 -0.5) :via (-12.0 8.0) (-11.0 8.0))
[22] - (grasp :object A :grasp 2)
[23] - (move-to (-9.0 8.0) :via (-5.0 8.0))
[24] - (ungrasp :object A)
[25] - (move-to (-17.0 -4.0) :via (-14.0 8.0) (-16.0 7.0) (-17.0 5.5))
```

Figure 8: A manipulation task generated automatically

5.4.2 Discussion and future extensions

We believe that the approach we propose will help to better understand the manipulation task planning problem. Because of its inherent combinatorial complexity, it can not be used for a great number of objects and placements. However, we think that it could be taken as a basis for studying relevant heuristics in order to tackle more realistic problems. We intend to study how the approach could be used together with a path planning module that manipulates only an approximated representation of the free space [13]: this could lead to consider object classes and regions in the workspace dedicated to object storage... The result would then be not a completely developed plan but a gross decomposition of the task into independent Pick-and-Place plan steps.

Another interesting development could be the study of a path planner module able to build incrementally a model of the free space. It could be invoked first to build a free space for an environment composed only by the robot and the obstacles; and then it could be requested to "add" an obstacle or to "adjoin" a body to the robot.

References

[1] R. Alami and H. Chochon. Programming of flexible assembly cells: task modeling and system integration. In *IEEE International Conference on Robotics and Automation, St Louis (USA)*, March 1985.

[2] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *IEEE International Conference on Robotics and Automation, San Francisco (USA)*, 1986.

[3] R.E. Fikes and N.J. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[4] C. Laugier and J. Troccaz. SHARP, a system for automatic programming of manipulation robots. In *Robotics Research: The Third International Symposium, O. Faugeras and G. Giralt (Eds), MIT Press, Cambridge, Massachusetts*, October 1985.

[5] J.P. Laumond and R. Alami. A new geometrical approach to manipulation task planning (1): the case of a circular robot amidst polygonal obstacles and a movable circular robot. In *Technical Report LAAS 88314, (submitted to IEEE Conf. on Robotics and Automation 1989)*, October 1988.

[6] J.P. Laumond and R. Alami. A new geometrical approach to manipulation task planning in robotics (2): a general solution based on motion planning. In *Technical Report LAAS (submitted to the First Canadian Conference on Computational Geometry)*, November 1988.

[7] T. Lozano-Perez and R.A. Brooks. *An approach to automatic robot programming.* A.I Memo 842, Artificial Intelligence Laboratory, MIT, April 1985.

[8] E. Mazer and T. Lozano-Perez. The structure of an interpreter for task-level robot programs. In *'87 International Conference on Advanced Robotics (ICAR), Versailles (France)*, October 1987.

[9] R. P. Paul. *Robot Manipulators: mathematics, programming and control.* MIT Press, 1981.

[10] E. Sacerdoti. *A Structure for Plans and Behavior.* Elsevier, North-Holland, New York, 1977.

[11] J. T. Schwartz and M. Sharir. On the piano mover II : general techniques for computing topological, properties of real algebraic manifolds. *Applied Math.*, 4, 1983.

[12] J.T. Schwartz and M. Sharir. On the piano movers' problem 3: coordinating the motion of several independent bodies: the special case of circular bodies amidst polygonal barriers. *International Journal of Robotics Research*, 2(3), 1983.

[13] T. Siméon. *Génération automatique de trajectoires sans collision et planification de tâches de manipulation en robotique.* Thèse de l'université Paul Sabatier, Laboratoire d'Automatique et d'Analyse des Systèmes (C.N.R.S.), Toulouse (France), Toulouse, France, January 1989.

[14] P. Tournassoud, T. Lozano-Perez, and E. Mazer. Regrasping. In *IEEE International Conference on Robotics and Automation, Raleigh (USA)*, 1987.

[15] G. Wilfong. Motion planning in the presence of movable obstacles. In *ACM Symp. on Computational Geometry*, 1988.

[16] C.K. Yap. *Algorithmic and geometric aspects of Robotics*, chapter Algorithmic motion planning. Lawrence Erlbaum Associates, London, 1987.

[17] C.K. Yap. *Coordinating the motion of several discs.* Robotics Report 16, Courant Institute of Math. Sciences, New York, 1984.