# Motion Planning for Multiple Moving Objects *

Yong K. Hwang
Sandia National Laboratories
Albuquerque, New Mexico 87185

## Abstract

We present a motion planner for multiple moving objects in two dimensions. The search for collision-free paths is performed in the composite configuration space of all the moving objects to guarantee a solution, and the efficiency of our planner is demonstrated with examples. Our motion planner can be characterized with a hierarchical, multi-resolution search of the configuration space along with a generate-and-test paradigm for solution paths. Because of the high dimensionality of the composite configuration space, our planner is most useful for cases with a small number of moving objects. Some of the potential applications are navigation of several mobile robots, and planning part motions for a multi-handed assembly operation.

## 1 Introduction

The multiple movers' problem is the problem of finding short and collision-free paths for many moving objects in an environment. Such a problem arises in path planning of multiple robots in factories or outdoor construction sites, or in assembly planning where the paths of many parts are needed to assemble them. Still another application is the path planning of manipulators working in the same workcell.

The complexity of the multiple movers' problem stems from the fact that the dimension of the search space, called the *composite* configuration space (composite Cspace or CCspace), is the sum of the degrees of freedom of all the moving objects. More degrees of freedom offer the benefit of potentially more solutions. Some sort of heuristic path planner can often be used to solve this problem. Developing an efficient and complete algorithm (that guarantees a solution for all cases), is, however, very difficult, since this involves an exhaustive search of the high-dimensional CCspace.

In our pursuit for an efficient *and* complete motion planner, we have developed an efficient search strategy called *Sandros* [3] that acts like a heuristic motion planner initially (solving easy problems quickly), and makes a graceful transition to an exhaustive search (guaranteeing a solution even for a hard problem). The exhaustive search is performed at a user-specified resolution, making our planner resolution complete. This planner was initially developed for motion planning of a single manipulator [3], and later extended to a single rigid object [9]. In this paper we present an application of the Sandros planner to multiple moving objects. Due to the general exponential worst-case nature of search algorithms, our algorithm is most efficient when applied to motion planning of a small number of robots.

In this paper, we use the term *robot* to mean objects whose motions we intend to control. The space of all poses of robots is referred as the composite configuration space. We define *dof* to be the total dimension of the CCspace. The next section reviews the previous work done on the multiple movers' problem, and Section 3 describes our motion planner. The performance of our planner is shown in Section 4, and the conclusions and future work are discussed in Section 5.

## 2 Previous Work

There has been a lot of work on the multiple movers' problem during the last 10 years. On the theoretical side, it has been shown to have a lower bound of PSPACE completeness and an upper bound that is exponential in the number of degrees of freedom. In [6], the planning of coordinated translational motion of rectangles (with a pattern of extrusions and indentations on their sides) in a rectangular boundary is shown to be PSPACE-hard by reducing the PSPACE-hard symbol transposition problem to it. This problem is later shown to be in PSPACE in [7], thus proving its PSPACE completeness. The single-exponential upper bound follows from the upper bound of the generalized motion planning problem [2].

In engineering applications where a practical solution is needed, many heuristic planners have been developed. They can be classified into centralized and distributed planning. In centralized planning, the motions of all robots are planned by a single decision maker. Centralized planning can be further divided into algorithms with or without priority among the robots. In certain applications, there are priority relations among the relative importance of the missions of robots. In other cases, the priority is artificially imposed to decompose the multiple movers' problem into a sequence of single-mover's problems. This, however,

*This work has been performed at Sandia National Laboratories and supported by the U.S. Department of Energy under Contract DE-AC04-94AL85000.

introduces the loss of optimality as well as completeness, i.e., the planned motion of a robot with a higher priority may preclude the motion of another robot with a lower priority. In [4], the obstacles in the composite configuration space are constructed to plan paths for two robots. In case there is a conflict between robots, the burden of collision avoidance is imposed on the robot with a lower priority. Motion planning of many square-shaped robots in the plane is presented in [1]. This algorithm has an application in an assembly cell, where multiple robots move on the ceiling of the cell, delivering parts and performing assembly operations. This algorithm tries to move as many robots as possible in straight lines, and minimizes the number of robots that have to make turns. Recently, game theory is used in [10] to plan the paths of multiple robots while optimizing multiple objectives. The robots are constrained to move on a roadmap such as the Voronoi diagram, and the concept of Nash equilibrium is used to schedule the motion of robots on the roadmap. A randomized approach is used to plan motions of multiple manipulators in [12]. This algorithm plans the motions of manipulators that move an object from one position to another. The path of the object is planned first, and the motion of each manipulator that moves the object along the planned path is computed by solving inverse kinematics. Note that the manipulators cannot carry the object outside their reachable space. The planner then assigns specific manipulators to carry the object at different portions of the path. Although this algorithm is not complete, it is efficient and solves many realistic problems.

A distributed planning approach is taken in [16] to plan paths for circular robots. Each robot plans its own path using the visibility graph, and when conflicts among the robots arise, it is reported to the central coordinator called the *blackboard*, which issues priorities based on the tasks and current state. Another decentralized approach for circular robots of different sizes is presented in [11]. Quadtree representation is used for path planning along with a Petri net formulation to resolve the conflicts among robots.

This paper presents a motion planner for multiple polygonal robots, each with 3 degrees of freedom, in a polygonal environment. Our algorithm plans the motion in the composite configuration space to guarantee a solution if one exists, and does not decompose the problem into a sequence of single-mover's problems. It has an important application in assembly planning, especially when an assembly requires multi-handed operations, i.e., those requiring coordinated movements of multiple parts. Our algorithm can also be used for multiple mobile robots working in a common environment such as in a construction site. We show the capability and efficiency of our algorithm with examples.

## 3 Algorithm

The best motion planning algorithm so far is exponential in the number of degrees of freedom [2]. If the number of degrees of freedom is low, a brute-force search algorithm is often sufficient. For problems with 5 dof or higher such as the multiple movers' problem, a brute-force search takes hours of computation time. The main feature of our algorithm is to find a solution path in a practical amount of time while maintaining the resolution completeness, i.e., guaranteeing a solution if it exists. This is exactly the motivation for the development of the *Sandros* search strategy, whose typical running time is in the range of a few minutes both for a manipulator [3] and for a rigid body [9]. In this paper we describe the extension of the Sandros methodology to the motion planning of multiple rigid bodies. The Sandros search method itself will only be sketched here; the details and the proof of resolution completeness can be found in [3, 9].

### 3.1 Overview of Sandros

The Sandros search strategy is basically a subgoal network approach [8], wherein a network of intermediate subgoals of the robot is maintained in a graph structure during the search. The network spans the free portions of the Cspace, implicitly generating the obstacles in the Cspace. The search for a collision-free path is done using the *generate-and-test* paradigm. A promising sequence of subgoals is generated and the existence of a collision-free path between successive subgoals is verified. Sandros uses a two-level planning hierarchy. The *global* planner maintains the subgoal network and produces sequences of subgoals. The local planner verifies the sequences until a solution is found. The global planner initially divides the Cspace into a small number of big cells, i.e., subgoals. If a solution is not found along any sequence of subgoals in the current network, some of the subgoals are refined into several smaller subgoals, expanding the subgoal network.

The success of the Sandros planner hinges on the subgoal generation and refinement process. For a manipulator, a subgoal takes the form of an affine space, i.e., the set generated by specifying the values of the first $n$ joints and letting the rest of the joints unrestricted. (An affine space is a linear subspace translated from the origin by a finite vector.) Subgoals are selected by configuring the first $n$ joint values so that the first $n$ links are maximally away from obstacles. The refinement of a subgoal of this form is done by specifying the value of the next unspecified joint. Figure 1 shows the subgoals at various refinement stages for a 3-dimensional Cspace. A refinement increases the number of subgoals and consequently the number of possible sequences to verify. To minimize the explosion of the subgoals, refinement is done only at promising portions of the Cspace. Figure 2 illustrates the Sandros search strategy applied to manipulator motion planning. Figure 2a shows a 2-link planar manipulator in a polygonal environment along with a collision-free motion between the start and goal configurations. Figure 2b shows the subgoals generated to solve this problem. The vertical-bar subgoals represent the good positions (positions maximally away from the obstacles) of the first link while ignoring the second link. They are generated when the whole Cspace (the initial subgoal) is refined. When a vertical-bar subgoal is refined, small-square subgoals are generated by selecting good positions of the second link while keeping the first joint at the vertical bar. The curves in Figure 2b show the traces of the local planner verifying various subgoal sequences. Figure 2c shows the solution path found and the Cspace obstacles.

Motion planning of a rigid body requires a different form of subgoals. While each joint value of a manipulator specifies the configurations of a subset of links, a rigid body has all of its degrees of freedom concentrated on a single body. For a manipulator, the goodness of a subgoal (which is typically a non-point set) can be measured by computing the distance to obstacles from the manipulator links whose positions are specified by the first $n$ joints. If we insist on having the affine-space form of subgoals for a rigid body, we could compute the volume swept by the robot while the configuration is varied in the subgoal, and get some sort of measure of the intersection volume between the swept volume and the obstacles. This has many drawbacks including the difficulty of computing the swept volume, numerical sensitivity of volume intersection, and large swept volumes of elongated objects giving poor measure of subgoal goodness.

A point subgoal with all the configuration variables specified allows direct measurement of subgoal goodness, namely, by computing the distance between the rigid body at that configuration and obstacles. However, we need to place point subgoals in a structured manner so as to facilitate the eventual exhaustive search if necessary. Due to these reasons, we have chosen as the form of subgoals for a rigid object a rectangloid cell and an associated point configuration contained in the cell. This representative point is then used by the local planner to verify the existence of a collision-free path between two subgoals. The point configuration is chosen using a random sampling in the cell until a collision-free configuration is found. If no collision-free point is found in a subgoal, it is not considered in computing a subgoal sequence for verification. It is kept, however, for later refinement. The refinement of a rectangloid subgoal is done by dividing it along its longest dimension into two rectangloids of an equal size. See Figure 3 for a snapshot of subgoals during planning.

If all subgoals are refined into small cells at a preset resolution, Sandros is equivalent to a brute-force search at that resolution. For most problems, however, a solution is found long before small subgoals are generated. In fact, easy problems with wide free space are often solved with a handful of subgoals. Note in Figure 2b that the majority of the Cspace is unexplored when the solution is found, and the subgoal refinement is done non-uniformly to minimize the number of subgoals. The local planner we have used is a potential-field based planner, which moves the robot toward a subgoal while maximizing the distance between the robot and the obstacles. The local planner declares success when the robot reaches any point of the target subgoal.

## 3.2 Extension to multiple rigid robots

We now describe the extension of the Sandros search strategy developed for a rigid body to multiple rigid bodies. Basically, this extension is done by forming the composite Cspace, i.e., by defining the configuration variables to be the concatenation of the configurations of all robots. Three things need to be determined to apply the Sandros search method: the form of subgoals, measuring goodness of subgoals, and the method of subgoal refinement.

We have considered three forms of subgoals for multiple rigid robots. One form of subgoals is to specify the configurations of some robots, while leaving the configurations of the rest of the robots unspecified. This is an affine-space form, although somewhat different from the subgoal form used for a manipulator. The goodness of a subgoal is then measured by the distances between the configuration-specified robots and obstacles. This method introduces a bias by ordering the robots for configuration specification, which implicitly assign priorities to robots.

The second form of subgoals is a rectangloid cell represented by a point in it, as done with a single rigid robot. The main advantage of this is that one can verify a sequence of subgoals in parallel. This is because the local planner *cannot* verify the existence of a collision-free path between two non-point subgoals. It needs at least one point subgoal, otherwise, the local planner does not have a specific configuration to initiate the path verification.. The drawback of this form is that a point subgoal is harder to reach than a subgoal with a finite volume. If the clearances among robots are large, this increases the computation time only marginally. This is because if a point in a rectangloid subgoal is not reached, then the subgoal will be divided into smaller cells, each represented by a point. The local planner is likely to find a collision-free path reaching one of the points if there is a collision-free path to the original subgoal. Therefore, having a point target in a subgoal might increase the refinement depth by 1 or 2. The speed up from a parallel computation is, however, much greater than the additional computation introduced by the increase in the refinement depth. If the spacing among robots are small, however, it is very hard to find collision-free robot configurations by random sampling.

Another option is to take rectangloid cells as subgoals, and the local planner will declare success when the robots reach any point in the rectangloid subgoal. This way, the local planner has a much higher success rate than when trying to reach a point. We have decided in favor of the third option. The refinement method is then the same as a single robot case, except the division of the rectangloid subgoal is performed in the composite Cspace.

## 4 Examples

We present several examples to show the characteristics of our planner. All computations times are the running times on a 100-MIPS workstation. Figure 4 shows the top view of two rigid people exchanging their positions. This problem is relatively easy, and it took our planner 9 seconds to solve. The local planner is invoked 4 times by the global planner. The next example is taken from [14] which was used to show that there is an assembly requiring an arbitrarily many hands to assemble it. Although his example has zero tolerance between the baseboard and the spikes, we added a finite tolerance to make our planner work. Our planner considers contacts as collisions. Overcoming this drawback is one of our future goals as discussed in the conclusion section. This example is solved in 5 seconds.

In Figure 6, we show an 18-dof problem where six mobile robots are trying to move to new locations in a room. It took our planner 5 minutes and 43 seconds to solve this problem. The planned motion is somewhat erratic since our planner tries to move the robots away from each other at all times. This problem would have been solved faster by decomposing it into a sequence of single mover's problems. Note, however, you need to reason about the order of moving robots. In this example, Robot 4 cannot reach its goal if it tries to move after Robot 3 and before Robot 6. Finally, Figure 7 shows a non-monotone assembly, i.e., one cannot assemble it by moving each part to its final configuration. The rectangular block must be put inside the U-shaped part before both of them can be put into the outside container. Our planner could not solve this problem in 20 minutes. The main reason is that it is very hard to generate the crucial configuration of the rectangular block inside the U-shaped part unless the contact conditions are used. In assembly-planning research, polyhedral convex cones (PCC) have been used to reason about local motion constraints [15]. We plan to use PCC to develop a global planner that generates extended motion. All the examples above show that our algorithm, in the current form, is most useful if the number of robots is small, they require simultaneous movements, and the space among robots is not small.

## 5 Conclusions and Future Work

We have developed a global motion planner for the multiple movers' problem in two dimensions. Unlike many existing planners for multiple robots which transform the problem into a sequence of single-mover's problems, we plan the motion in the composite configuration space of all robots. The extension of the Sandros search strategy has resulted in a motion planner that maintains the resolution completeness, while solving most realistic problems in minutes. Our planner is much more efficient than those planners that explicitly build the obstacles in the composite configuration space. One drawback of our algorithm is that since the motion is planned in the composite Cspace, our algorithm performs the best when the number of robots is small. Our planner has an important application in the assembly planning, especially for those containing multi-handed assembly, i.e., those requiring the simultaneous movements of more than one part. It is also useful for motion planning of multiple mobile robots working in a common environment.

We are currently implementing our planner for polyhedral robots, each with 6 degrees of freedom. The next research goal is to apply Sandros to multiple manipulators. The form of subgoals as well as the local planner need to be designed carefully to optimize the efficiency. Another research direction is to look into the motion planning with small tolerances among parts. In fact, most assemblies have very small or even negative tolerances. As addressed in the example in Figure 5, our planner needs a step size that is smaller than the tolerance between parts. Extension of our motion planning approach to such problems will require the use of polyhedral convex cones [5] to compute the feasible motion directions and magnitudes based on part contacts.

## References

[1] Buckley, S.J., "Fast Motion Planning for Multiple Moving Robots," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 322-326, Scottsdale, AZ 1989.

[2] Canny, J.F., *The Complexity of Robot Motion Planning*, The MIT Press, Cambridge, MA, 1988.

[3] Chen, P.C. and Hwang, Y.K., "SANDROS: A Motion Planner with Performance Proportional to Task Difficulty," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 2346-2353, Nice, France, 1992.

[4] Erdmann, M. and Lozano-Perez, T., "On Multiple Moving Objects," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1419-1424, San Francisco, California, 1986.

[5] Goldman, A.J. and Tucker, A.W., "Polyhedral convex cones," in Kuhn, H.W. and Tucker, A.W.

(eds.): *Linear Inequalities and Related Systems. Annals of Mathematics Studies 38*, Princeton, NJ: Princeton University Press, pp. 19-39.

[6] Hopcroft, J., Joseph, D. and Whiteside, S., "On the Movement of Robot Arms in 2-Dimensional Bounded Regions," *SIAM Journal of Computing*, vol. 14, no. 2, pp. 315-333, 1985.

[7] Hopcroft, J. and Wilfong, G.T., "Reducing Multiple Object Motion Planning to Graph Searching," *SIAM Journal of Computing*, vol. 15, no. 3, pp. 768-785, 1986.

[8] Hwang, Y.K. and Ahuja, N., "Gross Motion Planning - A Survey," *ACM Computing Surveys* vol 24, no 3, pp. 219-292, September 1992.

[9] Hwang, Y.K. and Chen, P.C., "A Heuristic and Complete Planner for the Classical Mover's Problem," in print, *Proceedings of IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May, 1995.

[10] LaValle, S. M. and Hutchinson, S. A., "Path selection and coordination for multiple robots via Nash equilibria," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1847-1852, San Diego, CA, 1994.

[11] Liu, Y.H., Kuroda, S., Naniwa, T., Noborio, H. and Arimoto, S., "A Practical Algorithm for Planning Collision-Free Coordinated Motion of Multiple Mobile Robots," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1427-1432, Scottsdale, AZ, 1989.

[12] Koga, Y, L. and Latombe, J.-C, "On multi-arm manipulation planning," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 945-952, San Diego, CA, 1994.

[13] Latombe, J.C., *Robot Motion Planning*, New York: Kluwer Academic Publishers, 1991.

[14] Natarajan, B. K., "On Planning Assemblies," *Proc. of the 4th ACM Symp. on Computational Geometry*, pp. 299-308, 1988.

[15] Wilson, R.H. and Matsui, T., "Partitioning an assembly for infinitesimal motions in translation and rotation," *Proc. of Int. Conf. on Intelligent Robots and Systems*, pp. 1311-1318, 1992

[16] Yeung, D.Y. and Bekey, G.A., "A decentralized approach to the motion planning problem for multiple mobile robots," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1779-1784, Raleigh, NC, 1987.

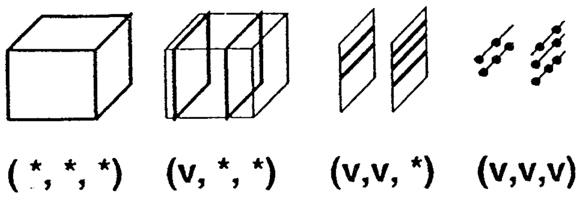

Figure 1. The affine-space form of subgoals in the 3-dimensional configuration space. It is used by the Sandros for motion planning of a manipulator.

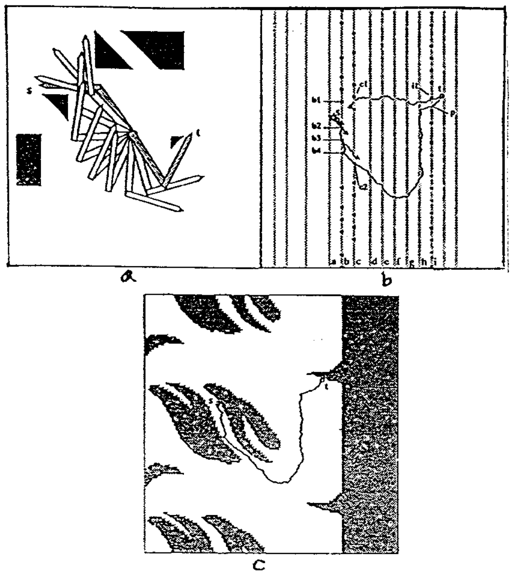

Figure 2. (a) A collision-free motion of a 2-link planar manipulator found by the Sandros planner. (b) The subgoals generated during the search process. (c) The solution path in the configuration space and the configuration-space obstacles (shaded regions).

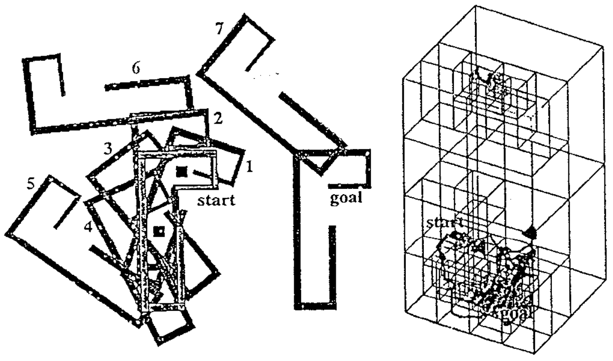

Figure 3. (a) Planning the motion of removing a clip away from two nails. (b) Rectangloid subgoals generated during the search by the Sandros planner.

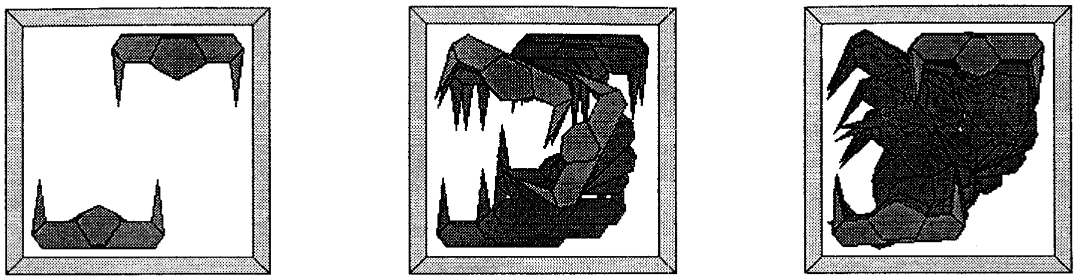
Figure 4. Two people exchanging their positions in a small room.

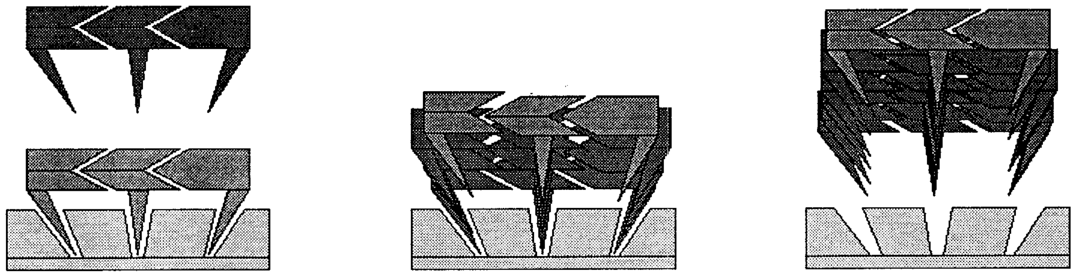
Figure 5. An assembly that requires simultaneous motions of three parts.

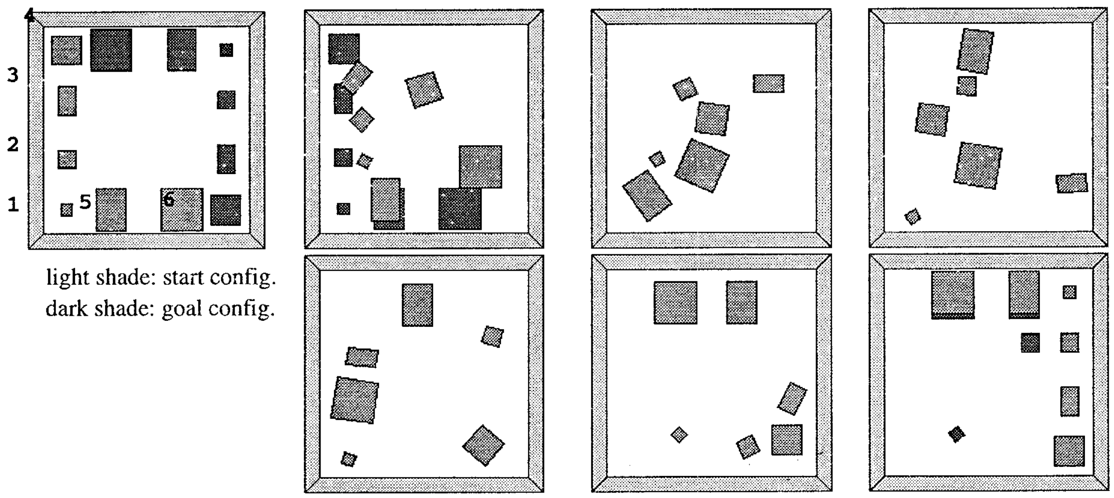

Figure 6. Six mobile robots switching their positions in a room. The motions are somewhat erratic since our planner tries to move robots away from each other.

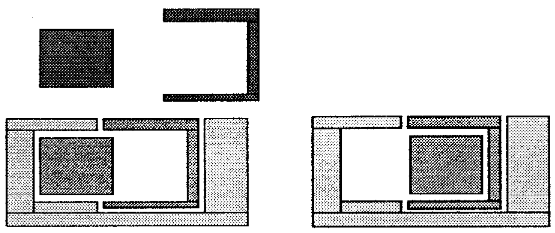
Figure 7. A non–monotone assembly.requiring the crucial configuration at right.