15th Triennial World Congress, Barcelona, Spain

ELSEVIER IFAC PUBLICATIONS
www.elsevier.com/locate/ifac

A MODEL FOR ASSEMBLY SEQUENCE PLANNING IN A MULTIROBOT ENVIRONMENT

C. Del Valle*, E. F. Camacho and M. Toro***

**Univ. Sevilla, Dept. Lenguajes y Sistemas Informáticos*
***Univ. Sevilla, Dept. Ingeniería de Sistemas y Automática*

Abstract: This paper presents a model for the selection of optimal assembly sequences for a product in multirobot systems. The objective of the plan is the minimization of the total assembly time (makespan). To meet this objective, the model takes into account, in addition to the assembly times and resources for each task, the times needed to change tools in the robots, and the delays due to the transportation of intermediate subassemblies between different machines. An A* algorithm that solves the problem is also presented, which starts from the *And/Or* graph for the product (compressed representation of all feasible assembly plans). *Copyright © 2002 IFAC*

Keywords: Assembly robots; flexible manufacturing systems; industrial robots; artificial intelligence; optimization problems; scheduling algorithms

1. INTRODUCTION

Assembly planning is a very important problem in the manufacturing of products. It involves the identification, selection and sequencing of assembly operations, stated as their effects on the parts. The identification of assembly operations usually leads to the set of all feasible assembly plans. The number of them grows exponentially with the number of parts, and depends on other factors, such as how the single parts are interconnected in the whole assembly, i.e. the structure of the graph of connections. In fact, this problem has been proved to be NP-complete in both the two-dimensional (Kavraki and Kolountzakis, 1995) and three-dimensional (Kavraki, *et al.*, 1995; Wilson, *et al.*, 1995) cases.

Two different approaches have been used in obtaining assembly plans. At first, interactive planners queried the user for geometric-reasoning information (Bourjault, 1984; De Fazio and Whitney, 1987). More recently, planners work automatically from a geometric and relational model of the assembly (Homem de Mello and Sanderson, 1991b) and from a CAD model and other non-geometric information (Ames, *et al.*, 1995; Romney, *et al.*, 1995).

Within this scope, the representation of assembly plans is an important issue. The use of *And/Or* graphs for this purpose (Homem de Mello and Sanderson, 1991a, b) is becoming one of the most standard ways of representing all possible assembly plans. It can be obtained by studying the opposite problem, that of disassembly, but maintaining the constraints of assembly. Most of automatic planners work with this strategy. The result is a representation which is adequate for a goal-directed approach. Moreover, Homem de Mello and Sanderson (1990) and Wolter (1992) showed that this structure is more efficient in most cases than other enumerative ones.

An optimum assembly plan is now sought, selected from the set of all feasible assembly plans. Two kinds of approaches have been used for choosing an optimal one. One, the more qualitative, uses rules in order to eliminate assembly plans that includes difficult tasks or awkward intermediate subassemblies. Another approach, the more quantitative, uses an evaluation function that computes the merit of assembly plans. There are various of these proposals in the book edited by Homem de Mello and Lee (1991), a monograph of the subject.

The criterion followed in this work is the minimization of the total assembly time (*makespan*) in the execution of the plan in a multirobot system. To meet this objective, we present a model which completes the one that was defined in a previous work (Del Valle and Camacho, 1996). The new model takes into account all factors which affect the makespan: an estimation of the duration of tasks; the resources used for them (robots and tools); the times needed for

changing tools in the robots; and the delays due to the transportation of intermediate subassemblies from one machine to another one.

An A* algorithm (Pearl, 1984) for obtaining the "best" assembly plan is also presented. The algorithm starts from the *And/Or* graph (compressed representation of all feasible assembly plans) and the information about each assembly task (robot and tool needed and assembly time), taking into account the delays referred in the last paragraph.

The rest of the paper is organized as follows: Section 2 describes the problem of assembly sequence planning, and Section 3 the model proposed. The A* algorithm is described in Section 4 and some of the results obtained are presented in Section 5. Some final remarks are made in the concluding section.

2. PROBLEM STATEMENT

The process of joining parts together to form a unit is known as assembly. The joining process results in the connection of one part with parts already assembled. A sub-assembly is a group of parts having the property of being able to be assembled independently of other parts of the product. An assembly plan is a set of assembly tasks with ordering amongst its elements. Each task consists of joining a set of subassemblies to give rise to an ever larger sub-assembly. An assembly sequence is an ordered sequence of the assembly tasks satisfying all the ordering constraints. Each assembly plan corresponds to one or more assembly sequences.

An *And/Or* graph is a representation of the set of all assembly plans possible for a product. The *Or* nodes correspond to sub-assemblies, the top node corresponds to the whole assembly, and the leaf nodes correspond to the individual parts. Each *And* node corresponds to the assembly task joining the subassemblies of its two final nodes producing the subassembly of its initial node. In the *And/Or* graph representation of assembly plans, an *And/Or* path whose top node is the *And/Or* graph top node and whose leaf nodes are the *And/Or* graph leaf nodes is associated to an assembly plan, and is referred to as an assembly tree. An important advantage of this representation, used in this work, is that the *And/Or* graph shows the independence of assembly tasks that can be executed in parallel. Figure 1 shows an example of this representation. And nodes are omitted.

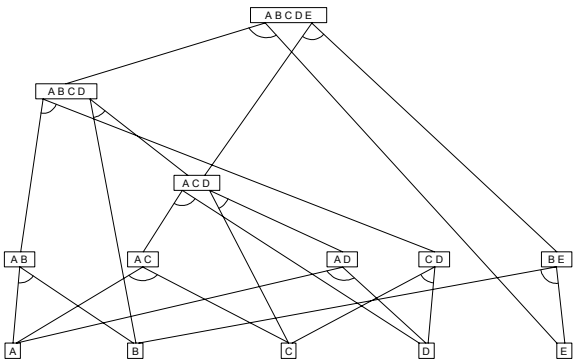

Fig. 1. The *And/Or* graph for the product ABCDE.

This work is centered on the problem of choosing the best assembly plan, that is, one of the *And/Or* trees of the *And/Or* graph. The majority of approaches used up to now make this selection in a planning phase in which neither the assembly system, nor how the assembly tasks within it will be materialized, is taken into account.

3. THE PLANNING MODEL

This work takes into account the physical realization of the assembly. It is assumed that the assembly tasks corresponding to the *And/Or* graph have been evaluated separately, in the sense of estimating the resources necessary for their realization (robots, tools, fixtures...) as well as their approximate duration times. For an *And/Or* graph with a large number of nodes this is not an easy task, and the help of an computer-aided system is necessary. The nodes corresponding to tasks which are not realizable as the adequate tools are not available are eliminated from the *And/Or* graph.

Another factor taken into account here, is the time necessary for changing the tools in the robots, which is of the same order as the execution time of the assembly tasks and therefore cannot be disregarded as in Parts manufacturing. $\Delta_{cht}(R,T,T')$ will denote the time needed for installing the tool T in the robot R if the tool T' was previously installed. Notice that any change of configuration in the robots can be modelled in this way.

Another question is the transportation of parts and subassemblies, that could affect the total assembly time. The proposed model supposes a well-dimensioned system, with a perfect planning when executing the assembly plan, so that, when a part would be required in a robot for executing an assembly operation, it will be present there. The same thing cannot be assured for an intermediate subassembly, because it could be built in a robot and required immediately in another one to form another subassembly. $\Delta_{mov}(SA,R,R')$ will denote the time needed for transporting the subassembly SA from robot R to robot R'.

The model considers only one combination *duration-robot-tool* for an assembly task. However, it can be extended easily when there are various ways for assembling a subassembly from the same components. It is enough to suppose that they corresponds to different assembly tasks, that is, we would add some alternative *And* nodes into the *And/Or* graph for the product.

With this model, the choice is not limited to the assembly plan, but also it can be specify when each task is to be carried out in order to minimize the

makespan (some tasks which could potentially be carried out in parallel have to be delayed because they need common resources).

The results derived from this model can be used in different stages of the whole planning process, from the design of the product and of the manufacturing system, to the final execution of the assembly plan.

4. ALGORITHM DESCRIPTION

As has been stated previously, the algorithm is centered on the choice of an assembly plan for a complete product in a multiple-robot system, where the resources necessary for carrying out each task represented in the *And/Or* graph (robots, tools...) appear as data, as well as the times necessary for their execution. As well as the choice of assembly plan, the execution orders for the tasks in each robot are specified by an analysis of their execution in parallel in the assembly system given.

Because of the set-up of the *And/Or* graph, the assembly problem can be studied, starting from the final situation and going towards the initial one.

The algorithm has two well-differentiated parts: one of them studies the sequential execution of assembly tasks, and the other solves the parallel execution of assembly tasks (the representation through the *And/Or* graph allows a natural study of this stage). This is actually the most complex section, because the execution of tasks on one side of the global assembly is not independent of the rest, and can influence the execution of tasks in the other part of assembly.

Heuristic functions based on the execution of tasks taking only from the part of the tree below the node, and the time remaining for the use of tools and robots (supposing the minimum number of tool changes, in order to maintain the algorithm as A*) has been used in order to expand the minimum number of nodes and avoid redundant nodes.

Because there is an upper limit to the makespan, the parallel algorithm does not need to finish when the best expected cost is higher than that limit.

The algorithm is used off-line to obtain an optimum first assembly plan. However, as the assembly process evolves, it can be used on-line to correct the changes which could have occurred during the assembly process, by pruning the *And/Or* graph of the subassemblies already performed.

4.1. Sequential Execution of Tasks

An A* algorithm to search for the global assembly plan can be implemented in the following way. Beginning with an initial node whose state represents the complete assembly realization, and therefore corresponds to the root node of the *And/Or* graph (complete assembly), all its possible successors are generated, whose states will represent the execution at the end of the assembly process of the tasks corresponding to the *And* nodes coming from the root node of the *And/Or* graph.

Two types of nodes may be generated, depending on the destination *Or* nodes of each chosen *And* node. If at least one of these *Or* nodes corresponds to an individual part, the assembly process will continue to be sequential, and the node resulting from the expansion may be treated as the initial node, where the node corresponding to the non-trivial sub-assembly will take the place of the root node.

If, on the other hand, the application of the task starts from two sub-assemblies each with various parts, in the resulting plan (or plans in general) the task arrangement is not totally specified (various possible sequences exist for each assembly plan), or tasks may be carried out in parallel. There is also an interdependence amongst the sub-assemblies, because they potentially use the same set of resources. The treatment of this type of nodes has therefore to be undertaken in a different way from those corresponding to sequential task execution, and this will connect with the second part of this algorithm.

The evaluation function used for the nodes generated in this part is

$$f(n) = g(n) + h(n) \qquad (1)$$

$g(n)$ being the time accumulated in the execution of tasks corresponding to the state of node n, including the delays in the necessary tool changes and in the transportation of intermediate subassemblies, and $h(n)$ being an optimistic estimation of the remaining time in which to complete the global process. ($h(n)$ should be a lower bound of the remaining time for the algorithm to be A^*.) Due to the fact that various different plans (and therefore different task sets which would complete the assembly process) may be reached from node n, a detailed study would be computationally costly, and therefore

$$h(n) = durMin \cdot \lfloor \log_2 \left(a(n) + 1 \right) \rfloor \qquad (2)$$

has been chosen, $a(n)$ being the number of tasks necessary to complete the assembly plan, and *durMin* the minimum duration of tasks. As can be seen, it is also impossible to determine the minimum number of tool changes without a detailed study, and therefore when estimating $h(n)$ it is assumed to be zero.

All the assembly trees (task precedence trees) are obtained for the "*parallel*" nodes, and are studied separately. The function $h(n)$ corresponding to each tree is defined in the following subsection.

4.2. Parallel Execution of Tasks

The objective of this part of the algorithm is to determine the total minimum time for the execution of the precedence trees obtained in the previous section. In

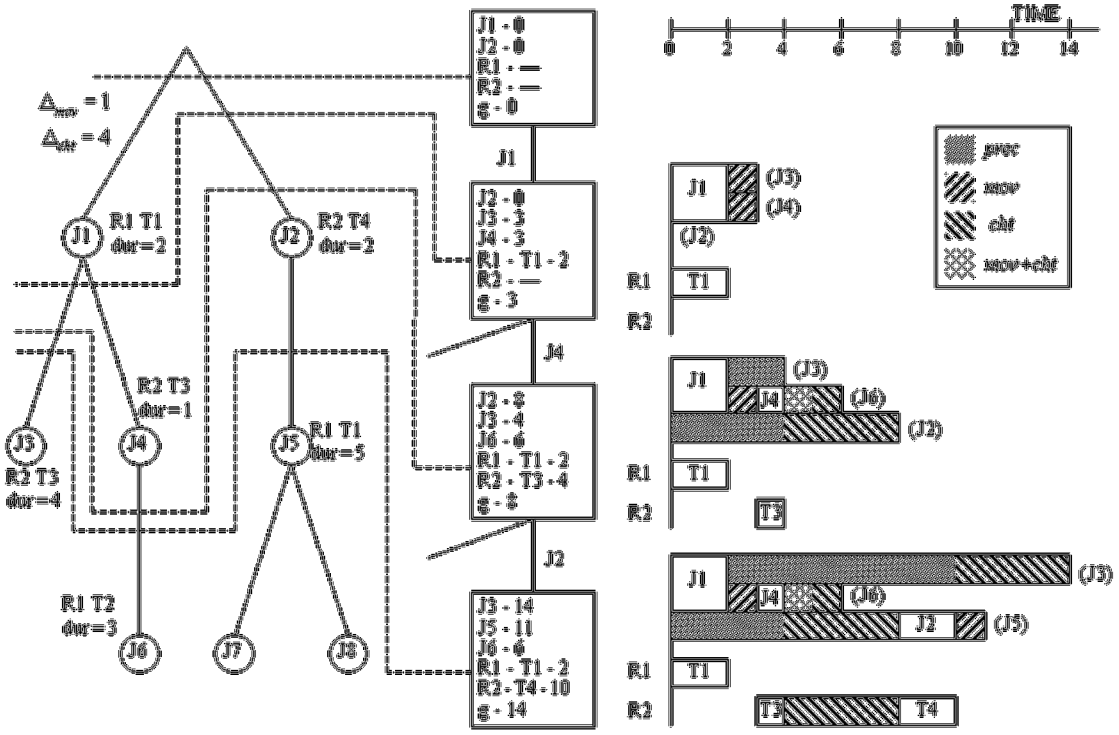

Fig. 2. A task precedence tree, some expansion nodes, and their corresponding Gantt charts.

order to do this, an A* algorithm is again used. The nodes of the expansion tree now present partial information about the execution of the assembly process. Concretely, at each expansion step only one assembly task is introduced, and its processing time will affect only one of the workstations, the same state being retained by the other workstations.

The state corresponding to a node of the expansion tree is represented by using the tasks available for introduction in the state of the next step, termed "*candidates*", and their earliest starting times, denoted *est*(J_i). At the same time, the last tool used is included for each robot, denoted *lastTool*(R_j), as well as the final time of use, denoted *lastTime*(R_j).

Figure 2 shows a task precedence tree, different expansion nodes and information about their corresponding states. It is also accompanied by the Gantt charts.

The evaluation function for the nodes obtained by this algorithm is similar to (1), being now $g(n)$ the largest of the earliest starting times of *cand*(n), the set of *candidates*, and the final times of the already finished in n without successors. The function $h(n)$ must be an optimistic estimation of the time remaining, taking into account the slacks between $g(n)$ and the different times describing the state of n. Two different heuristic functions have been defined in this work, taken from relaxed models of the problem in which some constraints have been disregarded.

The heuristic function h_1: *precedence of tasks*. It corresponds to an estimation of the time remaining if the interdependencies between different branches in the tree are not taken into account. It is looked at only in depth. It can be defined as follows:

$$h_1(n) = \max\left(0, \max_{J_i \in cand(n)} \left(h_1(J_i) - e(n, J_i)\right)\right) \quad (3)$$

where
$$e(n, J_i) = g(n) - est(n, J_i) \quad (4)$$

$$h_1(J) = dur(J) + \max_{J_i \in suc(J)} \left(h_1(J_i) + \tau_{mov}\left(J_i, J\right)\right) \quad (5)$$

$$\tau_{mov}\left(J_i, J\right) = \max\left(\tau\left(J_i, R(J), H(J)\right), \Delta_{mov}\left(sa(J_i), R(J_i), R(J)\right)\right) \quad (6)$$

$$\tau(J, R, T) = \max\left(0, \max_{T_i \in suc(T)} \left(\left(h_1(J_i) + \tau\left(J_i, R(J), T(J)\right) - h_1(J)\right)\right)\right) \quad (7)$$

In the above expressions, n is an expansion node, J is an assembly task, and $e(n, J)$ is the existing time slack. $R(J)$ and $T(J)$ are the robot and tool necessary for the execution of task J, and $dur(J)$ is its duration. $\tau(J, R, T)$ is the added delay, due to the fact that the tool T is being used by robot R in task J and *successors*, because of the necessary tool changes. The equation (7) defines $\tau(J, R, T)$ when $R \neq R(J)$. In the case $R=R(J)$, $\tau(J, R, T)$ is defined as $\Delta_{cht}(R, T(J), T)$ (that could be zero if $T=T(J)$). Finally, $\tau_{mov}(J, J')$ is the delay considering the possible transportation of the intermediate subassembly generated between the execution of J and J', and that of the possible change of tools.

Notice that $h_1(J)$ does not depend on the expansion nodes, and thus allows one to calculate a lower bound prior to using the A* algorithm.

The heuristic function h_2: *use of resources*. It corresponds to an estimation of the time needed if only the remaining usage times of the tools in each robot are

taken into account, further supposing the number of tool changes to be at a minimum. It can be defined as follows:

$$h_2(n) = \max_{robots}\left(h_2(n, R_i) - e(n, R_i)\right) \quad (8)$$

where $$e(n, R) = g(n) - lastTime(n, R) \quad (9)$$

and $h_2(n, R_i)$ is the minimum time of use of robot R_i without considering the task precedence constraints. If each tool is associated with only one robot, the calculation of $h_2(n, R)$ is equivalent to the traveling salesman problem, when considering the tools not yet used and an initial node corresponding to the last-used tool in the robot R:

$$h_2(n,R) = \left(\sum_{H_j \in R} \sum_{J_i \in cand(n)} h_2(J_i, T_j)\right) + \Sigma\left(n, \Delta_{cht}(R)\right) \quad (10)$$

with $h_2(J, T)$ the remaining time of usage of tool T by task J and its successors. The term $\Sigma\left(n, \Delta_{cht}(R)\right)$ refers to the time needed for the tool changes. In the usual case that tool-changing times do not depend on the type of tool, it can be calculated easily. Without any precedence information, an in order to maintain the admissibility of the heuristic, it must be supposed that the remaining tools will be installed only once.

Combination of heuristics. The heuristic functions h_1 and h_2 present two different effects in calculating $h(n)$. The estimation made from the first one is due to the most unfavorable candidate task. In the other hand, h_2 shows an additive effect, because of the uses of robots by all candidate tasks. Therefore, a new heuristic function can be defined from the combination of both, taking the most realistic estimation:

$$\max\left(h_1(n), h_2(n)\right) \quad (11)$$

Algorithm improvements. The use of A* algorithms presents some problems. The most important is the storage space that could be occupied. The algorithm has been adapted so that it uses a depth-first search periodically for finding a new solution whose value could be used for pruning the search tree.

Another improvement has been done by detecting symmetries, so that redundant nodes are avoided. The next definition shows it.

Definition: A task J_i is *compatible with* [including] task J_j if, on including this task at the following level, the start of J_i and that of its successors in the task precedence tree are not delayed.

This definition allows the number of expanded nodes to be minimized. The *candidate* tasks *compatible with* another task included in the next level will be included in successive levels.

The algorithm can be used in an off-line manner for obtaining an optimum initial solution for the assembly process. However, due on one hand to the flexibility for modifying the convergence criteria of the algorithm towards a not strictly optimum solution, and on the other to the fact that as the assembly process advances the resulting problem becomes smaller, the algorithm could be applicable on-line to modify either the plan or the initial sequence, in order to correct the variations with respect to the initial solution.

5. RESULTS

The algorithm has been tested in a variety of situations, considering different product structures (number of parts, number of connections between parts), different types of *And/Or* graphs (number of subassemblies, number of assembly tasks for each subassembly), and different assembly resources (number of robots, number of tools). The results in Tables 1-4 correspond to a hypothetical product of 30 parts, with 396 *Or* nodes and 764 *And* nodes in the *And/Or* graph. The number of linear sequences is about 10^{21}. The tables show the effect of having more or less resources for assembling the product in the performance of the algorithm. The results refer to 10 different combinations of durations and resources for assembly tasks. Apart from number of nodes visited and execution times, they show how many times the optimal solution was found by a depth-first movement (N-Pr), how many times the algorithm did not find the optimal solution in 30 seconds, when the available memory was exhausted (N-F), and the error rate.

6. CONCLUSIONS

A model for the selection of optimal assembly sequences for a product in a generic multirobot system has been presented. The objective of the plan is the minimization of the total assembly time. To meet this objective, the model takes into account, in addition to the assembly times and resources for each task, the times needed to change tools in the robots, and the delays due to the transportation of intermediate subassemblies between different workstations.

An A* algorithm is also presented, with some heuristic functions defined from relaxed models for the problem. The algorithm has been adapted in order to improve its computational efficiency. Some results have been presented, showing the influence of having different number of resources when executing the assembly plan.

REFERENCES

Ames, A.L., T.L. Calton, R.E. Jones, S.G. Kaufman, C.A. Laguna and R.H. Wilson (1995). Lessons Learned from a Second Generation Assembly Planning System. *Proc. 1995 IEEE Intl. Symp. on Assembly and Task Planning*, pp. 41-47.

Table 1 Results for 2 machines and 2 tools/machine

Heuristic	Nodes visited			Time (ms)			N-Pr	N-F	% Error
	Ave	Max	Min	Ave	Max	Min			
h_1	41492	89189	2520	19429	30590	180	4	6	0,990
h_2	9316	42775	32	1422	6420	0	5	0	0,000
$\max(h_1, h_2)$	16385	71585	32	4291	30050	0	4	1	0,248

Table 2 Results for 2 machines and 4 tools/machine

Heuristic	Nodes visited			Time (ms)			N-Pr	N-F	% Error
	Ave	Max	Min	Ave	Max	Min			
h_1	40093	56108	3020	25025	30930	710	2	8	2,648
h_2	5311	19398	267	790	4230	50	1	0	0,000
$\max(h_1, h_2)$	5078	19009	387	1143	4450	60	1	0	0,000

Table 3 Results for 4 machines and 2 tools/machine

Heuristic	Nodes visited			Time (ms)			N-Pr	N-F	% Error
	Ave	Max	Min	Ave	Max	Min			
h_1	16905	85540	32	2357	11700	0	1	0	0,000
h_2	2751	7195	32	263	710	0	2	0	0,000
$\max(h_1, h_2)$	804	2098	32	99	220	0	2	0	0,000

Table 4 Results for 4 machines and 4 tools/machine

Heuristic	Nodes visited			Time (ms)			N-Pr	N-F	% Error
	Ave	Max	Min	Ave	Max	Min			
h_1	22697	93376	32	6084	30530	0	4	1	0,302
h_2	1808	5907	128	197	660	0	1	0	0,000
$\max(h_1, h_2)$	1765	5907	32	278	980	0	2	0	0,000

Bourjault, A. (1984). *Contribution à une Approche Méthodologique de Assemblage Automatisé: Elaboration Automatique des Séquences Opératoires*. Thèse d'état, Université de Franche-Comté, Besançon, France.

De Fazio, T.L. and D.E. Whitney (1987). Simplified Generation of All Mechanical Assembly Sequences. *IEEE J. Robotics and Automat.*, **Vol. 3**, **No. 6**, pp. 640-658. Also, Corrections, **Vol. 4**, **No. 6**, pp. 705-708.

Del Valle, C. and E.F. Camacho (1996). Automatic Assembly Task Assignment for a Multirobot Environment. *Control Engineering Practice*, **Vol. 4**, pp. 915-921.

Homem de Mello, L.S. and A.C. Sanderson (1991a). Representations of Mechanical Assembly Sequences. *IEEE Trans. Robotics Automat*. **Vol. 7**, **No. 2**, pp. 211-227.

Homem de Mello, L.S. and A.C. Sanderson (1991b). A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences. *IEEE Trans. Robotics Automat*. **Vol. 7**, **No. 2**, pp. 228-240.

Homem de Mello, L.S. and A.C. Sanderson (1991c). Two Criteria for the Selection of Assembly Plans: Maximizing the Flexibility of Sequencing the Assembly Tasks and Minimizing the Assembly Time Through Parallel Execution of Assembly Tasks. *IEEE Trans. Robotics Automat*. **Vol. 7**, **No. 5**, pp. 626-633.

Homem de Mello, L.S. and S. Lee (eds.) *Computer-Aided Mechanical Assembly Planning*. Kluwer Academic Publishers.

Kavraki, L., J.C. Latombe and R.H. Wilson (1993). On the Complexity of Assembly Partitioning. *Information Processing Letters*. **Vol. 48**, pp. 229-235.

Kavraki, L. and M. Kolountzakis (1995). Partitioning a planar assembly into two connected parts is NP-complete. *Information Processing Letters*. **Vol. 55**, pp. 156-165.

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA, Addison-Wesley.

Romney, B., C. Godard, M. Goldwasser, G. Ramkumar (1995). An Efficient System for Geometric Assembly Sequence Generation and Evaluation. *Proc. 1995 ASME International Computers in Engineering Conference*, pp. 699-712.

Wilson, R.H., L. Kavraki, T. Lozano-Pérez and J.C. Latombe (1995). Two-Handed Assembly Sequencing. *International Journal of Robotic Research*. **Vol. 14**, pp. 335-350.

Wolter, J. (1992). A Combinatorial Analysis of Enumerative Data Structures for Assembly Planning. *Journal of Design and Manufacturing*. **Vol. 2, No. 2**, pp. 93-104.