# AUTOMATIC ASSEMBLY TASK ASSIGNMENT FOR A MULTIROBOT ENVIRONMENT

**C. Del Valle* and E.F. Camacho****

**Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, 41012 Sevilla, Spain (eduardo@etsii.us.es)*
***Departamento de Ingeniería de Sistemas y Automática, Universidad de Sevilla, 41012 Sevilla, Spain*

**Abstract:** This paper presents an algorithm $A^*$ for obtaining the "best" assembly plan for a product in a multirobot system. The algorithm takes into account, in addition to the assembly times, the times needed to change tools in the robots. The objective of the plan is the minimization of the makespan. To meet this objective, the algorithm starts from the And/Or graph (compressed representation of all feasible assembly plans) and the information on each assembly task (robot and tool needed, and assembly time).

**Keywords:** Flexible manufacturing systems; assembly robots; industrial robots; artificial intelligence; optimization problems; scheduling algorithms

## 1. INTRODUCTION

Automatic assembly is one of the areas of manufacturing that has not been fully developed in industry. This is mainly because robot control and off-line programming have not evolved as expected (Martensson, 1990), as result of the fact that the assembly process is much more complex than processes in other robotic applications and parts manufacturing. Therefore, until recently, industrial robots were used primarily in simple assembly applications. This situation is changing, however, and Flexible Assembly Systems have become a very important issue because of the need to produce small lots of products, but the degree of flexibility is still low (Boneschanscher, 1993).

There are different stages in the whole process of assembly, such as design of products and parts, design of fixtures, grasp selection, path planning, fine-motion planning, sensor integration, etc. One of the most important issues in the whole process is planning assembly tasks, whose optimality will have a significant effect on the final cost of the assembled product (Kusiak, 1990). The assembly planning problem 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, 1990, 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 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. A variety of criteria has been used for choosing an optimal one. For example, Wolter (1988) combines the ratings related to manipulability of subassemblies, fixture complexity, and the number of different directions from which operations are performed, to complete a plan. Criteria including the minimization of reorientation and fixture requirements were introduced by De Fazio *et al.* (1990). Henrioud (1989) proposed the aid of an expert about the operational and logistic complexity, and strategic advantages to select the best assembly tree (plan). Homem de Mello and Sanderson (1990) proposed assigning to the hyperarcs of the And/Or graph weights that depend on the complexity of the assembly tasks and on the stability of the intermediate sub-assemblies, and using generic search algorithms such as the AO$^*$ to obtain the best plan. Their proposal (1991c) of some optimization criteria based on maximizing the number of different assembly sequences encompassed by the assembly plan, and on maximizing the amount of parallelism (simultaneity) possible in the execution of the assembly tasks, is used in a heuristic search algorithm. In another way, an algorithm is proposed in (Holland, *et al.*, 1992) for a specific assembly cell, and for batches of products.

This paper presents an algorithm A$^*$ (Nilsson, 1980; Pearl, 1984) for obtaining the "best" assembly plan for a product in a multirobot system. The approach used here is that of Homem de Mello and Sanderson (1990; 1991c), but more detailed information for the assembly tasks is considered. The algorithm takes into account, in addition to the assembly times, the times needed to change tools in the robots. The objective of the plan is the minimization of the total assembly time (makespan). To meet this objective, 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).

The paper is organized as follows: Section 2 describes the problem of assembly-task assignment. The proposed algorithm is described in Section 3, and some of the results obtained are presented in Section 4. 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 sub-assemblies 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 sub-assemblies of its two final nodes producing the sub-assembly 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.

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

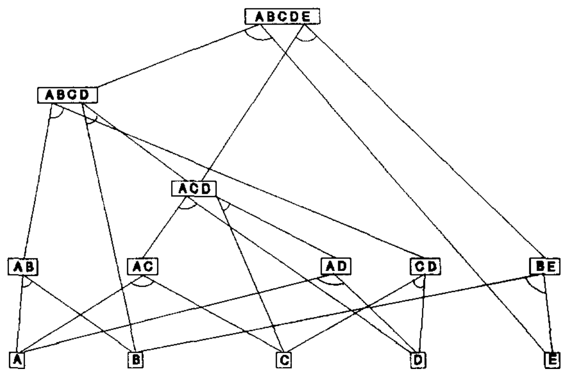

Fig. 1. The And/Or graph for the product ABCDE.

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.

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. These times should include an estimation for the times needed for other operations, such as transportation of parts and subassemblies. For an And/Or graph with a large number of nodes this is not an easy task, and the help of a 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 fact 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. Furthermore, the choice is not limited to the assembly plan, but also specifies 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 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.

## 3. 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 the assembly.

Heuristic functions based on the execution of tasks taken 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 $A^*$) have been used in order to expand the minimum number of nodes and avoid redundant nodes.

Because there is un 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. The optimization criteria can easily be changed, according to the particular needs of the application.

### *3.1. Sequential Execution of Tasks*

An algorithm $A^*$ 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 node 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

$$f(n) = g(n) + h(n), \tag{1}$$

$g(n)$ 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 $h(n)$ 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

$$h(n) = a(n) \cdot min(p_i) \tag{2}$$

has been chosen, $a(n)$ being the number of tasks necessary to complete the assembly plan, and $p_i$ the processing time of task $i$. 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.

## *3.2. 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 order to do this, an algorithm $A^*$ 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 one 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 $est(t_i)$. At the same time, the last tool used is included for each robot, as well as the final time of use.

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 *candidates(n)* and the final times of the already finished in $n$ without successors.

$$h(n) = \max(h_1(n), h_2(n)) \tag{3}$$

$h_1(n)$ = 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.

$h_2(n)$ = estimation of 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.

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 heuristic function $h_1(n)$ can be defined as follows:

$$h_1(n) = \max_{candidates(n)} \left( h_1'(n,J_i) - ft(n,J_i) \right) \tag{4}$$

where

$$ft(n,J) = g(n) - est(n,J) \tag{5}$$

$$h_1'(n,J) = h_1''(J) + \max_{robots} \left( \tau(J,R_i,last\_tool(R_i) - (est(n,J)-last\_time(R_i)),\ 0 \right) \tag{6}$$

$$h_1''(J) = p(J) + \max_{successors\ of\ J} \left( h_1''(J_i) + \tau(J_i,R(J),T(J)) \right). \tag{7}$$

In the above expressions, $n$ is an expansion node, $J$ is an assembly task, $last\_tool(R_i)$ and $last\_time(R_i)$ are the last tool used in robot $R_i$ and the time of last use respectively, and $(est(n,J)-last\_time(R_i))$ is the existing time slack. $R(J)$ and $T(J)$ are the robot and tool necessary for the execution of task $J$, and $p(J)$ is its processing time. $\tau(J,R,T)$ 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.

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.

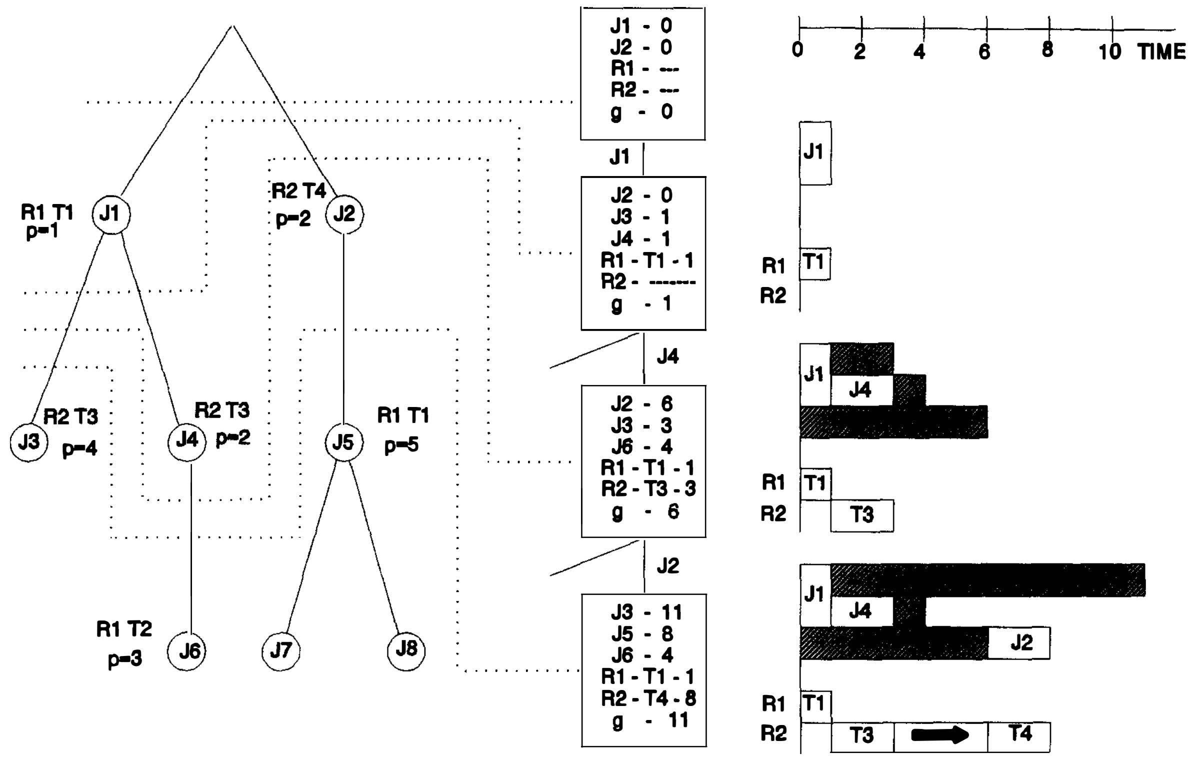

Fig. 2. A task precedence tree, some expansion nodes, and their corresponding Gantt charts.

$h_2$ can be defined as follows:

$$h_2(n) = \max_{robots} ( h_2'(n,R_i) - ft(n,R_i) ) \qquad (8)$$

where

$$ft(n,R_i) = g(n) - last\_time(R_i) \qquad (9)$$

and $h_2'(n,R_i)$ is the minimum time of use of robot $R_i$ without considering the task precedence constraints.

*First simplification*: Each tool is associated with only one robot. The calculation of $h_2'(n,R)$ is equivalent to the travelling salesman problem, when considering the tools not yet used and an initial node corresponding to the last-used tool in the robot $R$.

$$h_2'(n,R) = \sum_{T_i \in T(R)} \pi(T_i) + \textit{tool-change times} \qquad (10)$$

with $\pi(T_i)$ the remaining time of usage of tool $T$.

*Second simplification*: Tool-changing times do not depend on the type of tool.

$h_2(n)$ could be improved by using the earliest usable time of $R$ instead of using $ft(n,R_i)$. Notice that tasks not included in $n$ should be considered in this case.

*Definition*: A task $t_i$ is *compatible with* [including] task $t_j$ if, on including this task at the following level, the start of $t_i$ 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 *candidates* tasks *compatible with* another task included in the next level will be included in successive levels.

The expansion of a node is carried out by the algorithm shown in Fig. 3.

Notice that the algorithm can be extended to the case where there is more than one candidate tool for each assembly task. A list of candidate tools has to

```
Procedure Expand(n)
  Let $J = \{J_1,\ldots,J_k\}$ be the s t of candidates(n) and
    $EST = \{est_1,\ldots,est_k\}$ its earliest starting times.
  $est\_min = \min(est_i)$
  If there is just one task $J_i$ with $est_i = est\_min$
    Include in open a node whose state is that of n plus $J_i$
    If there are tasks not compatible with $J_i$
      Expand(n), restricted to $J' = \{J_m$, with $J_m$
                                    not compatible with $J_i\}$
    endif
  else
    Let $NTI_i$ be the number of tasks not compatible with $J_i$,
      and $NTI\_min = \min(NTI_i)$, for $est_i = est\_min$
    Include in open a node whose state is that of n plus $J_i$,
      with $J_i$ such that $NTI = NTI\_min$
    If $NTI\_min \neq 0$
      Expand(n), restricted to $J' = \{J_m$, with $J_m$
                                    not compatible with $J_i\}$
    endif
  endif
```

Fig. 3. Algorithm for the expansion of nodes.

be considered when expanding the nodes. A very simple heuristic function consisting of only considering the assembly times of the remaining tasks could be used. A more informed heuristic function would require a more complex algorithm.

## 4. 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 solution obtained for the assembly task assignment of the flashlight shown in Fig. 4 (Homem de Mello and Sanderson, 1990c) is shown in Fig. 5. The assembly environment was composed of two robots and two assembly tools per robot. The original complete And/Or graph contains 35 And nodes, 24 Or nodes and 37 possible assembly plans, and is not shown for the lack of space. The Gantt charts corresponding to the solution are shown in Fig. 6.

## 5. CONCLUSIONS

An A* algorithm for obtaining the optimum assembly plan for a multirobot environment has been presented. The algorithm minimizes the makespan of the assembly.

To apply the algorithm, possible assembly tasks should be specified by an And/Or graph. The algorithm needs the definition of the necessary tools and an estimation of the time required for each assembly operation.

The algorithm has been tested with problems of diverse complexity.

## 6. ACKNOWLEDGMENT

The authors would like to acknowledge CICYT for funding the work. The authors would also like to extend their thanks to the anonymous referees for their helpful suggestions.

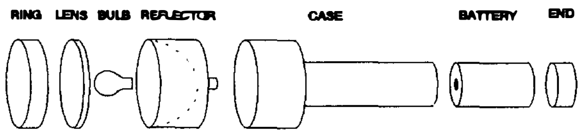

Fig. 4. Product example: a flashlight.

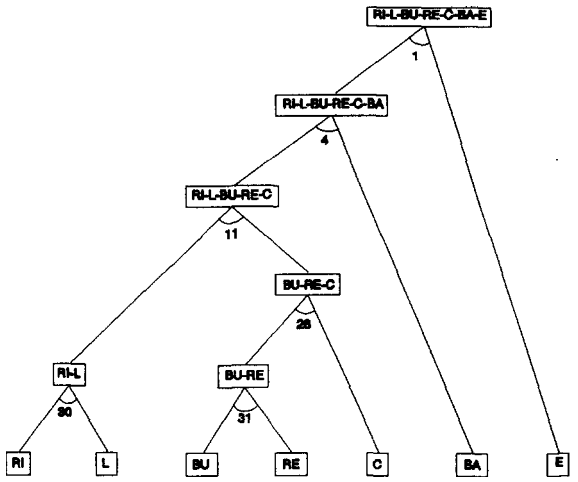

Fig. 5. Tree solution for the product example obtained from the And/Or graph.

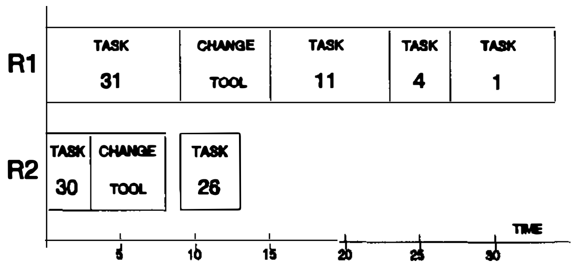

Fig. 6. Gantt charts for the tree solution in a two-robot environment.

## 7. 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.

Boneschanscher, N. (1993). *Plan Generation for Flexible Assembly Systems.* PhD thesis Delft University of Technology, Delft, The Netherlands.

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

De Fazio, T.L., T.E. Abell, G.P. Amblard, D.E. Whitney (1990). Computer-aided assembly sequence editing and choice: Editing criteria, bases, rules, and technique. *Proc. IEEE Int. Conf. Syst. Eng.*, pp. 416-422.

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.

Henrioud, J.M. (1989). *Contribution à la conceptualisation de l'assemblage automatisé: nouvelle approche en vue de détermination des processus d'assemblage*. Thèse d'état, Université de Franche-Comté, Besançon, France.

Holland, W. van, N. Boneschanscher and W.F. Bronsvoort (1992). Task Assignment in a Flexible Assembly Cell Using And/Or Graphs. *Proc. 23rd Int. Symp. Ind. Robots.* Barcelona, Spain, October 6-8, pp. 653-658, 642.

Homem de Mello, L.S. and A.C. Sanderson (1990). And/Or Graph Representation of Assembly Plans. *IEEE Trans. Robotics Automat.* **Vol. 6**, **No. 2**, pp. 188-199.

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.

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

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

Kusiak, A. (1990). *Intelligent Manufacturing Systems.* Prentice-Hall International Series in Industrial and Systems Engineering.

Martensson, N. (1990). Robot Ability for the 90's. *Proc. 21st Int. Symp. Ind. Robots.* Copenhagen, Denmark, October 23-25, 1990, pp. 193-198.

Nilsson, N.J. (1980). *Principles of Artificial Intelligence.* Chenanso Forks, NY: Tioga.

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. (1988). *On the automatic generation of plans for mechanical assembly.* Ph.D. thesis. Univ. of Michigan. Department of Computer, Information and Control Engineering, September 1988.

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