# On Selecting and Scheduling Assembly Plans Using Constraint Programming

Carmelo Del Valle[1], Antonio A. Márquez[2], Rafael M. Gasca[1], and Miguel Toro[1]

[1] Dept. Lenguajes y Sistemas Informáticos, Univ. Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
{carmelo, gasca, mtoro}@lsi.us.es
[2] Dept. Ingeniería Electrónica, Sistemas Informáticos y Automática, Univ. Huelva
Campus de La Rábida. Ctra. Huelva-Palos de la Frontera, 21071 Palos Fra. - Huelva
amarquez@uhu.es

**Abstract.** This work presents the application of Constraint Programming to the problem of selecting and sequencing assembly operations. The set of all feasible assembly plans for a single product is represented using an *And/Or* graph. This representation embodies some of the constraints involved in the planning problem, such as precedence of tasks, and the constraints due to the completion of a correct assembly plan. The work is focused on the selection of tasks and their optimal ordering, taking into account their execution in a generic multi-robot system. In order to include all different constraints of the problem, the *And/Or* graph representation is extended, so that links between nodes corresponding to assembly tasks are added, taking into account the resource constraints. The resultant problem is mapped to a Constraint Satisfaction Problem (CSP), and is solved using Constraint Programming, a powerful programming paradigm that is increasingly used to model and solve many hard real-life problems.

## 1 Introduction

Constraint Programming (CP) is a powerful programming paradigm that is increasingly used to model and solve many hard real-life problems. This work presents the application of CP to the problem of selecting and sequencing assembly tasks. This is a more difficult planning problem than others that have been tackled extensively in the literature using these techniques, such as the Job Shop Scheduling Problem (JSSP) and the Travelling Salesman Problem (TSP) [1] [2], since it involves not only the optimal arrangement of tasks, as in those problems, but also the selection of them from a set of alternatives. There has been little attention to this issue [3], so that the development of efficient CSP techniques considering it is of special interest.

The work is focused on the selection of tasks and their optimal ordering, taking into account their execution in a generic multi-robot system. The objective is the minimization of the total assembly time (makespan) [4], so that all factors that can

have an influence on it are taken into account: durations of tasks and use of resources among other additional elements.

The set of all feasible assembly plans for a single product is represented using an *And/Or* graph. This representation embodies some of the constraints involved in the planning problem, such as precedence of tasks, and the constraints due to the completion of a correct assembly plan. In order to include all the different constraints of the problem, the *And/Or* graph representation is extended, so that links between nodes corresponding to assembly tasks are added, taking into account the resource constraints. This representation allows a direct mapping of the problem to a Constraint Satisfaction Problem (CSP), in order to be solved using a CP approach.

The rest of the paper is organised as follows: Section 2 describes the problem of selecting and sequencing of assembly tasks, and Section 3 states the planning model defined to solve the problem. Section 4 shows the CSP model for the problem, and the extended *And/Or* graph illustrating the set of all constraints of the problem. Next section presents the CP-based method used for solving the problem, and some of the results obtained. Finally, some remarks are made in the concluding section.

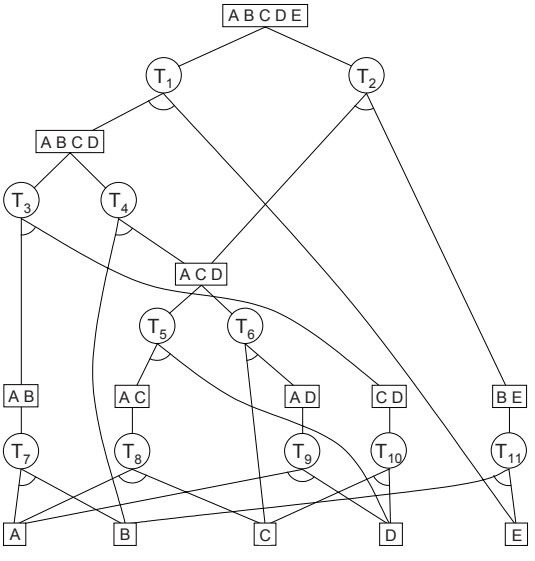


**Fig. 1.** *And/Or* for the product ABCDE

## 2 Assembly Sequence Planning

Assembly planning is a very important problem in the manufacturing of products. It involves the identification, selection and sequencing of assembly operations, specified by their effects on the parts. The identification of assembly operations has been tackled by analyzing the product structure [5] [6]. 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 within the whole assembly. In fact, this problem has been proved to be NP-complete [7].

An optimum assembly plan is now sought, selected from the set of all feasible assembly plans. Most approaches used for choosing an optimal one employ different kind of rules in order to eliminate assembly plans including difficult tasks or awkward intermediate subassemblies [8] [9]. The aim of this work is to obtain an optimal assembly plan, supposing its execution in a multi-robot system. To meet this objective, a previous estimation of durations and resources for tasks is needed.

*And/Or* graphs have been used to represent the set of all feasible assembly plans for a product [10]. In this representation, 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.

## 3 The Planning Model

The problem is focused on searching an optimal assembly sequence, an ordering of an assembly plan (one of the *And/Or* trees of the *And/Or* graph). This work starts from a previous estimation for the times and resources (robots, tools, fixtures...) needed for each assembly task in the *And/Or* graph. Another factor considered 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. $\Delta_{cht}(M, C, C')$ will denote the time needed for installing the tool $C$ in the robot (machine) $M$ if the tool $C'$ was previously installed. Notice that any change of configuration in the robots can be modeled in this way. Because of that, it will be used in the rest of the paper the more general terms machines and configurations.

Another issue is the transportation of parts and sub-assemblies, that could affect the total assembly time. Ideally, it would be supposed 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. But the same cannot be guaranteed for an intermediate subassembly, because it could be built in a robot and required immediately in another one to form another subassembly. $\Delta_{mov}(SA, M, M')$ will denote the time needed for transporting the subassembly $SA$ from machine $M$ to machine $M'$.

## 4 The CSP Model

The *And/Or* graph representation of assembly plans embodies the precedence constraints between assembly tasks, and the constraints related to the construction of a

correct assembly plan, i.e. a tree of the *And/Or* graph. An extension of this representation can be defined so that it includes all the constraints involved in the assembly planning problem described in Section 3, adding those constraints due to the use of resources, as it is shown in Figure 2. This section describes this extended representation and the associated CSP model.

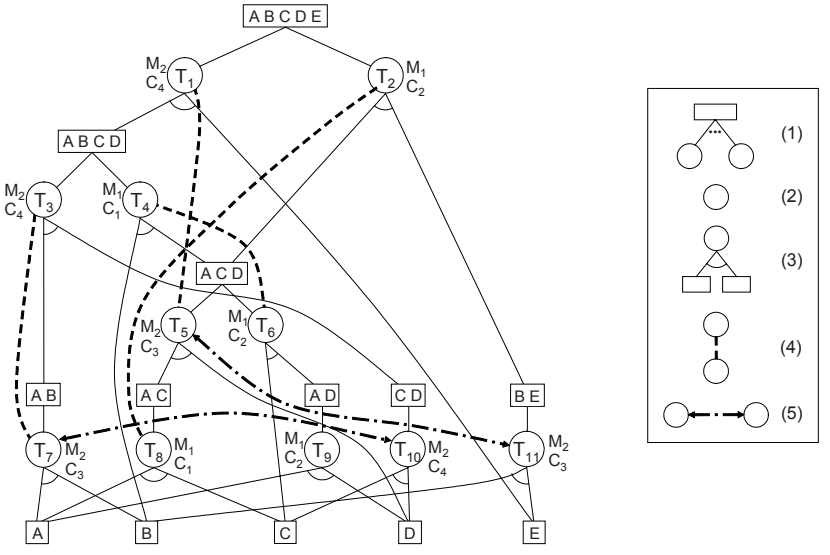


**Fig. 2.** The extended *And/Or* graph for the product ABCDE

Each node of the *And/Or* graph is associated to a set of variables or attributes. The previous section indicated those which could be considered constants: for each assembly task $T$ (*And* node), its duration, $dur(T)$, the machine used, $m(T)$, and the necessary configuration on it, $c(T)$; for the sub-assemblies (*Or* nodes), the delays related to the transportations between machines. Moreover, different temporal variables are considered for the two types of nodes: for each task $T$, its starting time, $t_i(T)$, and ending time, $t_f(T)$; for each sub-assembly $SA$, the time it was assembled, $t_{OR}(SA)$.

On the other hand, each task may not be present in the final solution. In order to build a correct solution, all the tasks selected must belong to one of the possible trees of the *And/Or* graph. Moreover, depending on the execution of some tasks or others, it will be formed some intermediate sub-assemblies and not another ones. So, an additional boolean variable is defined for each node in the *And/Or* graph, indicating if the node is selected for the solution, denoting as $s(T)$ and $s(SA)$ for a generic task $T$ and a generic sub-assembly $SA$ respectively.

In addition, since a sub-assembly car be built in different machines, depending on the selected task for its assembly, a new variable is defined in the model for the *Or* nodes, $m(SA)$ denoting the machine used for the assembly of $SA$.

All those variables form part of the CSP model proposed. Table 1 shows some indicative examples of the different types of constraints between the variables, corresponding to the extended *And/Or* graph of Figure 2. Constraints of type (1)

related the selection of tasks with that of sub-assemblies, expressed through the XOR operator, since one and only one alternative task can be selected to build a sub-assembly, if that sub-assembly takes part in the solution. Moreover, they define the constraints associated with the machine and assembly time of *Or* nodes, related to the tasks that can be selected. A special case is for the complete product and for the individual parts, which always will be part of the solution, so that the corresponding boolean variables *s* are *true*. Notice that, for the same reason, the specification of $t_{OR}$ for the individual parts is simpler (equals to zero).

**Table 1.** Set of constraints for the *And/Or* graph from Figure 2

| Typ | Constraints |
|---|---|
| (1) | $s(ABCDE) = s(A) = s(B) = s(C) = s(D) = s(E) = true$ |
| | $s(ABCDE) \Rightarrow \left(s(T_1)\ XOR\ s(T_2)\right)$ |
| | $s(T_1) \Rightarrow \left(m(ABCDE) = M_2 \wedge t_{OR}(ABCDE) = t_f(T_1)\right)$ |
| | $s(T_2) \Rightarrow \left(m(ABCDE) = M_1 \wedge t_{OR}(ABCDE) = t_f(T_2)\right)$ |
| | $s(ABCD) \Rightarrow \left(s(T_3)\ XOR\ s(T_4)\right) \quad \wedge \quad \neg s(ABCD) \Rightarrow \quad \left(\neg s(T_3) \wedge \neg s(T_4)\right)$ |
| | $s(T_3) \Rightarrow \left(m(ABCD) = M_2 \wedge t_{OR}(ABCD) = t_f(T_3)\right)$ |
| | $\vdots$ |
| | $t_{OR}(A) = t_{OR}(B) = t_{OR}(C) = t_{OR}(D) = t_{OR}(E) = 0$ |
| (2) | $s(T_1) \Rightarrow t_f(T_1) = t_i(T_1) + dur(T_1)$ |
| | $\vdots$ |
| | $s(T_{10}) \Rightarrow t_f(T_{10}) = t_i(T_{10}) + dur(T_{10})$ |
| (3) | $s(T_1) \Rightarrow \left(s(ABCD) \wedge t_i(T_1) \geq t_{OR}(ABCD) + \Delta_{mov}\left(ABCD, m(ABCD), M_2\right)\right)$ |
| | $s(T_1) \Rightarrow t_i(T_1) \geq t_{OR}(E)$ |
| | $\vdots$ |
| | $s(T_{11}) \Rightarrow t_i(T_{11}) \geq t_{OR}(B)$ |
| | $s(T_{11}) \Rightarrow t_i(T_{11}) \geq t_{OR}(E)$ |
| (4) | $\left(s(T_5) \wedge s(T_1)\right) \Rightarrow t_i(T_1) \geq t_f(T_5) + \Delta_{cht}(M_2, C_3, C_4)$ |
| | $\left(s(T_6) \wedge s(T_4)\right) \Rightarrow t_i(T_4) \geq t_f(T_6) + \Delta_{cht}(M_1, C_2, C_1)$ |
| | $\left(s(T_7) \wedge s(T_3)\right) \Rightarrow t_i(T_3) \geq t_f(T_7) + \Delta_{cht}(M_2, C_3, C_4)$ |
| | $\left(s(T_8) \wedge s(T_2)\right) \Rightarrow t_i(T_2) \geq t_f(T_8) + \Delta_{cht}(M_1, C_1, C_2)$ |
| (5) | $\left(s(T_5) \wedge s(T_{11})\right) \Rightarrow \left(t_i(T_5) \geq t_f(T_{11}) \vee t_i(T_{11}) \geq t_f(T_5)\right)$ |
| | $\left(s(T_7) \wedge s(T_{10})\right) \Rightarrow \left(\left(t_i(T_7) \geq t_f(T_{10}) + \Delta_{cht}(M_2, C_4, C_3)\right) \vee \left(t_i(T_{10}) \geq t_f(T_7) + \Delta_{cht}(M_2, C_3, C_4)\right)\right)$ |
| | minimize $t_{OR}(ABCDE)$ |

The constraints of type (2) consider the durations of tasks.

The constraints of type (3) include the precedence between start times of *And* nodes and assembly times of *Or* nodes, and the possible delays due to the transportation of sub-assemblies between different machines. Moreover, they related

the selection of sub-assemblies to that of the tasks using them to build another bigger one.

The constraints of type (4) are due to the delay needed for a change of configuration in a machine between the executions of tasks using the same machine with precedence constraints among them. Notice that it is only needed relating each task to its closest predecessor one in the *And/Or* graph that uses the same machine. Furthermore, when both tasks use the same configuration, the resulting constraint is superfluous and can be eliminated. A new type of link between *And* nodes has been added in order to represent this kind of constraints in the extended *And/Or* graph of Figure 2.

Finally, the constraints of type (5) express the two possible orders when executing two tasks with no precedence constraints among them that use the same machine, maybe with a change of configuration in the machine. Also, a new type of link between *And* nodes has been added in order to represent this kind of constraints in the extended *And/Or* graph of Figure 2.

Notice that the combinatorial character of the problem is due to the constraints of types (1) and (5), corresponding to the selection of alternative tasks, and to the use of shared resources from tasks that are not related through precedence constraints.

## 5 Solving with Constraint Programming

Usually, a combined method of searching (instantiating variables) and constraint propagation (reducing the domains of variables) is used in CP for solving a CSP. Both approaches can ignore the specific problem to be solved, so that a non-deterministic method can be used. However, the algorithm becomes more efficient if a good knowledge of the problem is put in both the search and constraint propagation methods.

The algorithm used here is based on backtracking. It maintains a set of tasks, all belonging to the same tree of the *And/Or* graph, the constraints related to these tasks, and a list of *Or* nodes to expand. The algorithm makes two different types of expansion steps when exploring the *And/Or* graph, one for the *Or* nodes and the other for the *And* nodes. The expansion of an *Or* node results in the selection of an alternative task, that is created and its constraints posted (and removed when backtracking) dynamically. The expansion of an *And* node results in adding the *Or* nodes to the open list. When the algorithm has completed a tree of tasks, i.e. the list of *Or* nodes is empty, it uses an *edge-finder* propagator to increase propagation, provided that there are no more tasks that can use the resources.

The algorithm has been implemented using ILOG Solver and Scheduler [11], a C++ library for Constraint Programming and scheduling problems. Table 2 shows the results obtained for a hypothetical product of 30 parts, whose *And/Or* graph contains 396 *Or* nodes and 764 *And* nodes. There are about $10^{21}$ possible linear arrangements of the tasks forming a legal solution. The results come from 11 different combinations of durations and resources for the tasks, for a system with 2 machines and 2 possible configurations for each machine. The problems were tested without using a specific method for selecting the next *And* and *Or* nodes from the alternatives in the algorithm, denoted as "-" heuristic, and using a heuristic based on the number of trees below the node. For the selection of the *Or* node, the node with the least number of trees was

selected, so that propagation of precedence constraints could have a better effect. For the *And* node, that of the greatest number of trees, so that there are more alternatives where to find the optimal solution. The table shows, for each combination of heuristics, how many problems were solved in less time than others, the average time, and the number of problems that the time used by the algorithm deviated more than the double from the average time.

**Table 2.** Comparative results when using heuristics in the exploration of the *And/Or* graph

| Heuristic (*Or* / *And*) | # Best result | | Average | # 2·Ave | |
|---|---|---|---|---|---|
| - / - | 0 | (0.0%) | 258.23 | 1 | (9.1%) |
| - / max #trees | 0 | (0.0%) | 253.36 | 2 | (18.2%) |
| min #trees / - | 4 | (36.4%) | 249.99 | 1 | (9.1%) |
| min #trees / max #trees | 7 | (63.6%) | 244.75 | 2 | (18.2%) |

## 6 Conclusions

A CSP model has been presented for the selection of optimal assembly plans in a multi-robot system. The different factors that could be an influence in minimizing the total assembly time in a generic assembly system have been taken into account.

The CSP is represented through an extended *And/Or* graph, so that all the different constraints of the problem are symbolized through different types of links between the nodes of the graph, whose attributes are the variables of the CSP. Some results from a CP-based algorithm are shown in the paper.

## References


[1] Caseau, Y., Laburthe, F.: Improving Branch and Bound for Jobshop Scheduling with Constraint Propagation. Proceedings of the 8th Franco-Japanese 4th Franco-Chinese Conference CCS'95 (1995)

[2] Esquirol, P., Fargier, H., Lopez, P., Schiex, T.: Constraint programming. Belgian Journal of Operations Research, Statistics and Computer Sciences (1996)

[3] Beck, J.C., Fox, M.S.: Constraint-directed techniques for scheduling alternative activities. Artificial Intelligence, 121 (2000) 211-250

[4] Del Valle, C., Camacho, E.F.: Automatic Assembly Task Assignment for a Multirobot Environment. Control Eng. Practice, Vol. 4, No. 7 (1996) 915-921

[5] Homem de Mello, L.S., Sanderson, A.C.: A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences. IEEE Trans. Robotic and Automation, Vol 7 (2) (1991) 228-240

[6] Calton, T.L.: Advancing design-for-assembly. The next generation in assembly planning. Proc. 1999 IEEE Int. Symp. Assembly and Task Planning (1999) 57-62

[7] Wilson, R.H., Kavraki, L., Lozano-Pérez, T., Latombe, J.C.: Two-Handed Assembly Sequencing. Int. Journal Robotic Research. Vol. 14 (1995) 335-350
[8] Homem de Mello, L.S., Lee, S. (eds.): Computer-Aided Mechanical Assembly Planning. Kluwer Academic Publishers (1991)
[9] Goldwasser, M.H., Motwani, R.: Complexity measures for assembly sequences. Int. Journal of Computational Geometry and Applications, 9 (1999) 371-418
[10] Homem de Mello, L.S., Sanderson, A.C.: And/Or Graph Representation of Assembly Plans. IEEE Trans. Robotics Automation. Vol. 6, No. 2 (1990) 188-199
[11] ILOG, France, http://www.ilog.fr/