RALB - A Heuristic Algorithm for Design and Balancing of Robotic Assembly Lines

Jacob Rubinovitz, Joseph Bukchin - Submitted by Ehud Lenz (1)
Received on January 7,1993

Summary

This paper describes an heuristic approach for design and balancing of a robotic assembly line. The objective of RALB (Robotic Assembly Line Balancing) algorithm is to balance the assembly line, by allocating equal amount of work to the stations on the line, while assigning the most efficient robot type, out of several different types of robots available for the assembly task, to each workstation, and minimizing the number of workstations and robots used. RALB uses heuristics to limit and guide a Branch and Bound frontier search, thus leading to solution of very large or difficult problems. A recommendation of the optimal set of heuristic rules is made based on results of extensive testing of RALB with a variety of assembly problems.

Keywords: Robotic Assembly, Line Balancing, Frontier Search

1. Introduction

Assembly plays a central role in the manufacturing of most products and systems which we use. Parts that are individually designed, made, and inspected, must be assembled into the final product. The final product must function correctly, and the entire assembly process must be executed efficiently and without error.

The economic importance of assembly as a manufacturing process has led to extensive efforts for improving the efficiency and cost effectiveness of assembly operations. Assembly production lines can be manually operated, automated, or of mixed design. Manual assembly is characterized by high labor costs, while automated assembly *requires very* high investment in dedicated equipment. In both cases, there is a substantial motivation to perform these operations as efficiently as possible, A common way to achieve efficiency in an assembly process, is to perform it on an assembly line. This system consists of multiple workstations, in which part of the assembly task is accomplished, as the product is moved from station to station. One of the key advantages of using such a system for manual assembly lines is specialization of labor. By giving each worker a limited set of tasks which are performed repeatedly, fast learning takes place, and the worker is able to perform the tasks at a faster rate and more consistently. In automated systems, there is a need for specialization of tooling and feeding devices, which directs the division of the assembly task into a *series* of workstations. For both types of systems to be efficient, the line must be balanced, which means that the individual processes must be allocated to workstations in such a way that the total assembly time required at each assembly station is approximately the same. If such balance cannot be achieved, inefficiencies in the form of idle time at stations, or temporary blocking or starving of stations will result,

Traditionally, production has been organized in flow or assembly lines for situations of mass production - long term production of large quantifies of the same products. Such situations warranted the investment in dedicated assembly equipment, line conveyance equipment, and in setting and balancing of the assembly line.

The customer demand for greater product variety, and the effort of manufacturing companies to maintain a competitive advantage by introducing new products, along with fast changing technologies, have created a new reality in which product life-cycles are shorter, quantities are smaller, and variety is larger. This requirement for flexibility of response to customer demands, led to the development of Flexible Assembly Systems (FAS), in which the assembly robot plays a key role. The need of flexible systems which do not sacrifice high productivity is further emphasized by Milberg and Schmidt (10), leading them to the assumption that flexible assembly cells and flexible assembly lines will gain considerable significance.

Assembly is a relatively difficult task for robotic implementation. Part of this difficulty stems from the large variety of assembled parts, and the resulting need for multiple feeders, grippers, and other mechanical interfaces which may limit system flexibility. Flexible assembly systems can be designed in several configurations:

a) An assembly cell with a single robot.
b) A cell with multiple robots which share tools, resources, or cooperate in a common work space.
c) A robotic assembly line.

While all configurations warrant flexibility, the robotic assembly line is the only configuration which provides for specialization of tooling, grippers, and fixtures, and can deliver faster assembly rates, with better system efficiency. Robotic assembly lines have an advantage over dedicated assembly lines for medium and small production volumes. In such an environment, several robots with different capabilities are available on the assembly floor, and they have to be reassigned in an optimal way to the assembly task each time a new product islaunched.

This paper deals with the design and balancing of such robotic assembly line. There are two main objectives in designing this assembly system, which have to be satisfied simultaneously. One is to achieve an optimal balance of the assembly line, while satisfying the required production rate. The other is to allocate the most efficient robotic equipment to each station, taking advantage of the specialization of equipment provided by the assembly line configuration.

2. Robotic Assembly Line Balancing

Methods for assembly line balancing have been developed in the past. These methods were designed for balancing manual assembly lines. A detailed survey of exact methods for solving the simple assembly line balancing (SALB) problem, and their underlying assumptions as formulated for manual assembly**,** is given by Baybars (1). There are two main assumptions common to most SALB problem formulations:

a) Task element times are deterministic and independent of the equipment, operator, or station to which the task is assigned.
b) A task can be performed at each station if technological precedence constraints are satisfied, and if the system cycle time is not exceeded by assigning the task to a workstation.

Robotic assembly lines operate under a different set of constraints and assumptions. A different robot, tooling, and assembly equipment may be used at each station, placing constraints on tasks assignment to a given station. Task performance times may be dependent on the specific robot and equipment selected for the task.

Another difference between manual and robotic assembly lines is in the amount of variation of task times. In manual assembly, there is a considerable variation in actual task performance from the standard time estimate used for line balancing. As a result, achieving a "perfect" line balance is of theoretical importance only, and good balances, achieved with use of heuristic methods would suffice in practice. However, in a robotic assembly line there is almost no variation from the established work pace and task performance times. As a result, any imbalance of the line and idle time at certain workstations will actually reduce system performance.

These specific problems of robotic assembly lines have been mostly ignored by researchers. Graves and Holmes (6) presented an optimal algorithm for equipment selection and task assignment for multi-product assembly systems. The system is an assembly line, to which activities and equipment have to be assigned, satisfying the annual production rate, and the assembly precedence constraints. The objective of this work is to minimize total cost, which is composed of fixed equipment and tooling costs, and of variable equipment usage and gripper exchange costs. This objective may be different from an objective of maximum efficiency of the assembly line which is achieved by line balancing. The input to the algorithm consists of equipment costs, task times on different equipment types, and a set of possible assembly sequences which is derived from a Liaison Diagram as suggested by DeFazio and Whitney (5). The algorithm finds the minimum cost configuration for a mixed-model assembly line. The main limitation of the algorithm is in using a single assembly sequence for each model. Since most assembled products may be assembled using several alternative sequences, this algorithm finds only a local optimum, and does not take advantage of the assembly task flexibility described by the task precedence diagram. As a result, idle times at each station are not minimized.

3. Robotic Assembly Line Balancing (RALB) Problem Formulation

The robotic assembly system considered in this work consists of a number of assembly robots of different types. Part of these robots will be used to create an assembly line for a single product. This is an assembly environment in which production runs are relatively short, and hence the flexibility provided by general-purpose assembly robots is required. When a new product is

launched, RALB algorithms can be used to design the new assembly line, by planning which robot type will be assigned to each station on the line, and by balancing the line. Each robot type may be capable to perform all or part of the assembly tasks. Deterministic task times are assumed for completion of each task work element by each robot. However, the performance times of the same work element by different robots may vary, as a result of different robot performance capabilities.

The possible product assembly sequences are defined by a precedence diagram, which describes all the technological precedence constraints resulting from the product design.

Following is a summary of the RALB model assumptions:

1) The precedence relation between assembly task elements is known, and it does not change.
2) The assembly task is composed of basic task elements which cannot be further subdivided.
3) Task element times are deterministic.
4) Task element times are dependent on the robot selected to perform the task element.
5) A task element can be performed at any station of the assembly line, provided that the robot selected for this station can do the task, and that precedence relations are observed.
6) A single robot is assigned to each station on the line.
7) The line is balanced for a single product.
8) Material handling, loading and unloading times are negligible or included in the task element times.
9) Set-up and tool changing times are negligible or included in the task clement times.
10) The purchase cost of the robots is not considered by RALB, as the different types of robots are assumed to be available.

The objective of the RALB algorithm is to find a line balance which will minimize the number of stations on the assembly line, for a given assembly production rate. This assembly production rate determines the maximum cycle time for the system. The output from RALB specifies the assembly stations, the type of robot assigned to each station, and the list of task elements performed at each station.

4. RALB Algorithm

The RALB (Robotic Assembly Line Balancing) method is based on a Branch-and-Bound algorithm using the Frontier Search method. Branch and Bound methods have been used extensively for solving NP complete problems such as machine scheduling (3), or assembly line balancing (7,9). These methods are in general computationally intensive, in both computer storage and processing time requirements. A trade-off between the storage and processing time requirements is possible, based on the type of search used by the Branch and Bound algorithm (Frontier Search v.s Depth Search).

The main advantage of the Branch and Bound method is the ease and flexibility in which the algorithm can be extended to accommodate new constraints, or relax some of the basic model assumptions, such as the 10 assumptions of the RALB model. Another advantage of this type of algorithms is the ease of incorporation of heuristic rules, which reduce the search space for large and complex problems. Such rules were used, for example, to reduce the search space for large and complex problems in a multiple solutions approach to the single-model assembly line balancing problem using the MUST algorithm (4).

The Branch and Bound approach has been used to solve the single model line balancing problem by several researchers, using different search strategies, such as best bud search (11), frontier search (12), or newest node search (8). In all these algorithms, the definition of good lower bounds, combined with rules for resolving ties during the search process, was crucial for finding computationally feasible solutions for large problems.

The RALB algorithm uses the Branch and Bound approach suggested for solving single model assembly line balancing problems, with modifications necessary to solve a system with multiple and different robot types. It uses the Frontier Search method (12), while incorporating search limiting rules. Some of these rules were adapted from the works of Jackson (7), Mansoor and Yadin (9), and Dar-EI and Rubinovitch (4). The algorithm creates a search tree, by assigning work-elements and robots to stations, until an optimal solution is reached. The main stages in this search process are:

1) Creation of the first (top) level of the search tree. At this level, each node contains a work element without preceding elements (based on the precedence matrix), along with a robot capable of performing this work elements. Such nodes are created for all feasible work-element and robot combinations.

2) Selection of a node to be extended. This selection is based on a calculation of a lower bound on the number of assembly stations for each node, as follows:

$$FLB = \begin{cases} N + (T^*/C) & \text{if } F_{in} = 1 \\ N + ((T^* - S_c)/C) & \text{if } F_{in} = 0 \end{cases}$$

where: $T^* = \sum_j \text{Min}(t_{ij})$

t_{ij} - the time of work-clement i when performed by robot j
N - number of stations already assigned
C - cycle time for which the assembly line is being balanced
$\mathcal{O}$ - a set of work elements which were not yet assigned to stations at the current stage
S_c - The slack time at the current station
F_{in} - An index with the value of I if the current station is closed (fully assigned) and a new station will have to be open at the next stage, or 0 otherwise.

It can be noted that T* is the total sum of the shortest times of the work elements which still have to be assigned, being in fact a lower bound on the work content still to be assigned. In the case of a station which has not been completely assigned (F. = 0), V is adjusted by the slack value at the station S., assuming that activities which will be assigned to this open station will fully utilize the slack time. Since the value of FLB is a real number, and the lower bound on the number of stations has to be integer, the actual lower bound is the smallest integer value which is larger than FLB:

$$LB = [FLB]^+$$

The node with the smallest LB value is extended. However, in a case of a tie, additional rules are applied for selection of the node to be extended, as follows:

(a) select the node which is at the lowest level in the search tree.
(b) If there are several nodes at the same lowest level, select the one with the lowest FLB value.

3) Node extension. The selected node is extended to new nodes, by adding one different feasible work element to each new node. A feasible element is an element with all predecessors already assigned, and ~ < S,. For extension of an open station (Fi. = 0), the robot j assigned to this station is used. If the extension is from a closed station (Fi. = 1), new nodes are open for each feasible work element and each robot which can perform it.
4) If the node which has been extended contains all the assembly work elements, the optimal solution has been found. Otherwise, return to stage 2.

5. RALB problem parameters and complexity measures

The RALB algorithm frontier search strategy ensures that the first solution found is an optimal solution. However, for large and complex problems, the memory space required to store all the active nodes (nodes which are candidates for extension) may exceed the available computer memory, or the computation time required to reach a solution may become prohibitively long, In order to solve such real-world problems, heuristic rules which limit the search space need to be introduced. Good understanding of the RALB problem parameters and structure, and the resulting computational complexity of a given problem, is essential for the effective design of heuristic rules.

In this work, two measures were introduced to measure the computational complexity of a given RALB problem. Both measures are independent of the computer type used. These measures are:

1) Total Nodes - a measure of the time required to reach a solution, counting the total number of nodes generated in the search process. The total number of nodes is directly proportional to the number of algorithm iterations, and hence, the computation time.
2) Open Nodes - a measure of the memory storage space required. Only the open nodes are stored at each stage of the algorithm, and the Open Nodes measure contains the maximum number of Open Nodes which had to be stored simultaneously during the search for solution.

The following parameters were used to characterize the RALB problem complexity:

1) Assembly Production Rate - this production rate is used to calculate the cycle time of the assembly line. For lower the cycle times, a larger number of stations will be required, and it may be more difficult to achieve good balance and high line efficiency.
2) Problem Size - this is measured by both the number of work elements in the assembly task, and the number of different robot types available.
3) Assembly Flexibility - this is the measure of flexibility of the assembly sequence, as measured by the technological constraints given by the precedence diagram, or Fp,, (9). For a given precedence matrix P, such that:

$$P_{ij} = \begin{cases} 1 & \text{if task i precedes task j} \\ 0 & \text{otherwise} \end{cases}, \qquad F_{Ratio} = (2*Z)/(k*(k-1))$$

where Z is the number of zeroes in P_{ij}, and k is the number of assembly work elements.

4) Robotic Assembly System Flexibility - this flexibility is a result of different alternatives available by using different robot types, each with different task performance times. This measure of system flexibility RF can be defined as follows: For a given matrix T_ describing the work element times % for n work elements and m robots, ~ is the time to perform work element i by robot j. If robot j cannot perform work element i, % is set equal to infinity. RF is defined as:

RF = I - M/(n*(m-1)) where M is the number of elements set to infinity in T_{nm}

The results of evaluating the RALB algorithm performance for the different task complexity parameters are shown in Figures 1 through 4.

As for the Assembly Production Rate parameter, it can be seen that there is a very significant growth in the computation time required in a narrow zone in which a slight change of the cycle time causes the addition of a station in **order to achieve a balance.** In other areas, the computational effort is low. Also, the computational effort is reduced for longer cycle times (Fig. 1).

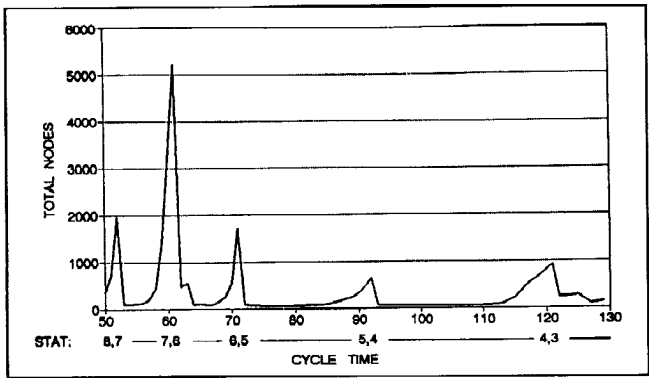


Figure 1: Total Nodes as a Function of Cycle Time

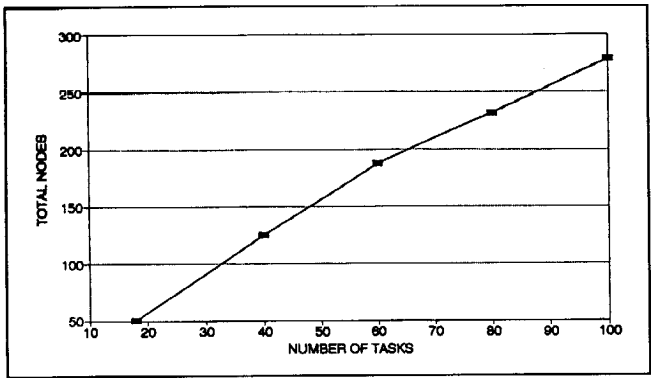


Figure 2: Number of Total Nodes as Function of Problem Size (measured by task work-elements)

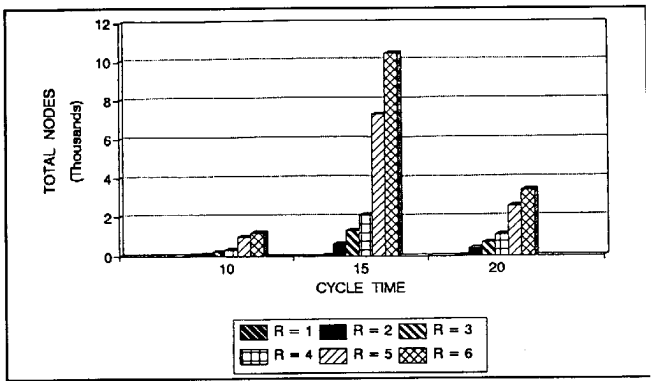
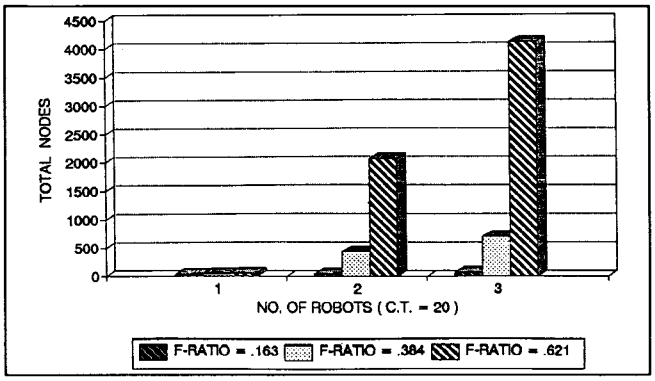


Figure 3: Total Number of Nodes as Function of Problem Size (expressed by the number of robots)

As for the Problem Size parameter, there is a linear growth in the number of Total Nodes (computation time) with a growth of number of tasks (Fig. 2). The same effect was also observed for memory requirements (Open Nodes). There is a significant growth in the number of Total Nodes with growth in the number of robot types (Fig. 3).



Fgure 4: Total Number of Nodes as Function of the F,._

Assembly Flexibility (F_{Ratio} affects both the computation time (Total Nodes) and required storage space (Open Nodes), and there is an increase in both measures for unconstrained assembly networks with high F_{Ratio} values. However, while computation time is a] most not affected by the number of robot types for heavily constrained networks (low F_{Ratio} values), it grows at a very significant rate when more robot types are used along with high F_{Ratio} values (Fig.4).

No clear trend in problem complexity growth as a result of changes of the RF parameter could be indicated. However, in general less Open Nodes (storage space) were required to solve similar problems with a lower RF value (low Robotic Assembly System Flexibility).

6. Heuristic Rule for Solving Large RALB Problems

In order to solve large and complex RALB problems, a heuristic rule which limits the search space was introduced. This rule eliminates nodes which have relatively low probability to lead to an optimal solution. The rule is activated by limiting the number of levels of the search tree for which open nodes are kept, This heuristic rule can be tuned dynamically, as to the number of levels kept, taking into account the available storage space, and problem complexity.

In the initial stages of the frontier search, the optimal algorithm extends the nodes with the lowest bounds (LB), assigning new activities to assembly stations resulting from these nodes. As a result, the lower bound value of these nodes increases, and the algorithm needs to return and extend nodes previously abandoned due to a relatively high LB value. For example, let's assume that in an assembly with 40 work-elements, 38 already have been assigned by the algorithm while following a search path, leading to a FLB value of 7. 1, Le 8 assembly stations (LB=8). At this time, a node at a much higher level in the search tree has a FLB value of 6.9 (7 stations), but represents an assignment of only 12 work elements. The optimal algorithm will extend this node, although there is a high probability that soon enough its FLB value will exceed 7. 1, and this search path would be abandoned again.

Hence, the heuristic rule used is a limit on the number of levels for which open nodes are kept. Large values of this limit can provide better and closer to optimum solutions, but at the expense of longer search time, and larger memory space for storage of the open nodes. Using a heuristic parameter of 1 results in a fast depth search, in which the node with the lowest bound is extended at each level.

Application of this rule removed the storage space constraints, and allowed the solution of difficult problems which could not be solved by the optimal algorithm. These problems included up to 100 work elements, and up to 3 different robot types, at various F_{Ratio} values and different Robotic Assembly System Flexibility (RF) values.

7. Testing of the Heuristic Rule Modes of Operation

The way in which nodes are selected for extension, and the extension process, do not affect the final solution when an optimal algorithm is used. In the extension process, the node with the smallest LB value is extended, but if several nodes with the same LB value exist, the following tie resolving order is used:

(a) First, select the node which is at the lowest level in the search tree.

(b) If there are several nodes at the same lowest level, select the one with the lowest FLB value.

While extending a node and assigning work elements, the lexicographic rule is used to avoid creation of identical nodes in the optimal algorithm. This rule assigns activities to a single station in an ascending order of work-element numbers.

When applying the heuristic rule in a search process, a lot of nodes are being eliminated from the search. As a result, using the lexicographic rule, may affect the search direction and quality of results. Also, reversing the order of tie resolution may affect the quality of solutions.

To establish the recommended mode of operation for the heuristic algorithm, a test program was developed, to compare four modes of operation, with and without the lexicographic rule, and using two different priority orders for tie-resolving, as summarized in Table 1.

Table I
Programs for Testing of Heuristic Rule Implementation Modes

	With Lexicographic rule	Without Lexicographic rule
Node Level ↓ FLB	P1	P2
FLB ↓ Node Level	P3	P4

For each of the four test programs, 9 different problems were solved. These problems included 60, 80 and 100 work elements. For all the problems, three different robot types were available on the assembly line. Each problem was balanced for three cycle times, of 80, 140 and 200 time units. The quality of solution for each problem under the four operation mode programs was ranked from I to 4, by giving the highest rank of I to the mode which reached solution with the minimal number of stations (and robots) used. If the number of stations was the same for several modes, the tie was resolved by looking at the total nodes and maximum number of open nodes needed to reach the solution, as measures of program solution speed and memory requirements. The total ranking for all the problems under the four operation modes is summarized in Table II.

Table II
Rating of Heuristic Rule Implementation

	With Lexicographic rule	Without Lexicographic rule
Node Level ↓ FLB	P1 33.0	P2 20.5
FLB ↓ Node Level	P3 24.0	P4 12.5

From these test results, it is clear that operation mode P4 provides the best solutions. It was also shown that an optimum solution was reached for five out of the nine problems, by comparing the final number of stations to the problem lowest bounds.

8. Conclusion

An algorithm (RALB) for the design and balancing of a robotic assembly line, with different robot types and capabilities, has been developed and tested for different problem characteristics. Extensive testing was conducted to recommend the best operation modes in implementing the heuristic rule.

RALB is an efficient algorithm, capable of solving difficult robotic line assembly problems in a reasonable computation time, and within storage space constraints. The solution technique used (Branch and Bound Frontier Search) allows for easy inclusion of real-life constraints, such as limiting the number of robots of each type which are available, etc. As such, RALB can be an useful tool for solution of line balancing problems using robots for flexible assembly in medium and small batch sizes.

References

1. Baybars, L, 1986, "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem", *Management Science*, 32, 8, p.909.

2 Bukchin, J., 1991, "Design and Balancing of a Robotic Assembly Line", unpublished M.Sc: Research Thesis, Technion, Israel Institute of Technology, (In Hebrew)

3 Charlton, J. M.,and Death,C. C., 1969, "A General Method for Machine Scheduling", *Int. J. Prod. Res.*, 7, p.207.

4. Dar-El, E. M. and Rubinovitch,Y., 1979, "Must- A Multiple Solutions Technique for Balancing Single Model Assembly Lines", *Management Science*, 25, 11, p. 1105.

5. DeFazio, T.L. and D.E. Whitney, 1986, "Simplified Generation of All Mechanical Assembly Sequences", C.S. Draper Laboratory Report No. P-2709, Cambridge, Massachusetts.

6. Graves, S. C. and Holmes, C. A., 1987, "Equipment Selection and Task Assignment for Multiproduct Assembly System Design", WP #1895-87, Massachusetts Institute of Technology, Cambridge, Massachusetts.

7. Jackson, J. R., 1956, 'A Computing Procedure for a Line Balancing Problem", *Management Science*, 2, p.261.

8. Johnson, R. V., 1988, "Optimally Balancing Large Assembly Lines With FABLE", *Management Science*, 34, p.240.

9. Mansoor, E. M. and Yadin, M., 1971, "On the Problem of Assembly Line Balancing", *Developments in Operations Research*, edited by B. Avi-Itzhak, Gordon & Breach, New York, p.361.

10. Milberg, J. and Schmidt, M., 1990, "Flexible Assembly Systems -Opportunities and Challenge for Economic Production", Annals of the CIRP, 39, 1, p. 5.

11. Nevins, A. J., 1972, "Assembly Line Balancing Using Best Bud Search", *Management Science*, 18, 9, p.530.

12. Van Assche, F. and Herroelen W.S., 1979, "An Optimal Procedure for the Single-Model Deterministic Assembly Line Balancing Problem", *European J. Oper. Res., 3, 2, p. 142.*