ANT COLONY ALGORITHM FOR NEAR-OPTIMAL ARRANGEMENT OF LINEAR MACHINE LAYOUT

Apostolis Papadimitriou,George Georgoulas, Chrysostomos Stylios[1] and Peter Groumpos

Laboratory for Automation & Robotics, University of Patras, 26500, Patras, Greece
Tel. +30 2610997293 Fax +302160997309 Email: georgoul@ee.upatras.gr

[1]Dept. of Informatics and Telecommunications Technology, TEI of Epirus,
47100 Artas, Epirus, Greece Tel. +30261050330 Email:stylios@teleinfom.teiep.grr

Abstract: Ant Colony System (ACS) is a meta-heuristic methodology, inspired by the behaviour of natural ants, which has already been applied to a numerous combinatorial problems. Flexible Manufacturing Systems cope with multi-product, usually small sized production. In this research work we investigate and apply an Ant Colony Optimization algorithm, which arranges the machines of a production line so that to minimize the total amount of backward flows. The experimental results show that a near optimal solution can be found exploiting only a small portion of the feasible solution space. Thus, the proposed algorithm indicates that it is a promising method, which can be applied to complex shop floor configuration for a generalised layout.

Keywords: ant colony optimization, machine layout problem, ant colony system.

1. INTRODUCTION

Ant colonies are distributed systems that in spite of the simplicity of their individual units, they present a highly structured social organization and they can accomplish complex tasks, which in some cases far exceed the individual capacities of a single ant. One of the most surprising behavioural patterns exhibited by ants is the ability of certain ant species to find the "shortest paths". Biologists have shown experimentally that this ability is possible by exploiting communication capabilities based only on pheromones, an odorous chemical substance that ants may deposit and smell. A well known class of ant algorithm known as "Ant Colony Optimization" (ACO) (Dorigo and Stutzle, 2004) has been inspired by the behaviour of natural ants.

Ant algorithms were introduced in early 1990's (Dorigo *et al.*, 1991a; Dorigo *et al.*, 1991b) and they were combined with a multi-agent approach for difficult combinatorial optimization problems. These problems usually belong to the class of *NP*-hard problems (problems for which the computational effort needed to find optimal solutions increases exponentially with problem size). Examples of combinatorial problems are, the shortest-paths problems, as well as many other real-world problems like finding the minimum cost plan to deliver goods to customers, the optimal assignment of employees to tasks to be performed, the best routing scheme for data packets in the Internet, the optimal sequence of jobs which have to be processed in a production line, the allocation of flight crews to airplanes, and many more (Dorigo and Stutzle 2004; Dorigo *et al.,* 1999).

Two classes of algorithms are available for the solution of combinatorial optimization problems: exact and approximate algorithms. *Exact algorithms* try to find optimal solutions. Despite their recent successes, for many *NP*-hard problems the performance of exact algorithms is not satisfactory and their applicability is often limited to rather small instances. *Approximate algorithms* trade optimality for efficiency. Their main advantage is that, in practice, they often find reasonably good solutions in a very short time. Algorithms of this type are loosely called *heuristics* (the term heuristics comes from a Greek word that means "to discover", from which also comes the famous Archimedes' exclamation "eureka") (Rich and Knight, 1991). Recently, a new class of algorithms emerged, called *metaheuristics*. A meta-heuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems. The use of meta-heuristics has significantly increased the ability of finding very high-quality solutions for hard combinatorial optimization problems in a reasonable time.

A particularly successful meta-heuristic is the ACO meta-heuristic. Algorithms that fit into the ACO meta-heuristic framework are called ACO algorithms. ACO is characterized as being a distributed, stochastic search method based on the indirect communication of "artificial" ants, mediated by pheromone trails. The pheromone trails in ACO serve as distributed numerical information used by the ants to probabilistically construct solutions for the problem under consideration. The ants modify the pheromone trails during the algorithm's execution giving a clue to other ants about their search experience. The ACO meta-heuristic defines the way the solution is constructed and the pheromone trails are modified by enriching artificial ants with capabilities that are not present in real ants. A large number of different ACO algorithms have been proposed and can be found in the literature. The main differences between the various implementations of algorithm concern the techniques used to control the search process. These extensions include among others: *Elitist AS* (Dorigo 1992; Dorigo, *et al.*, 1991a, Dorigo, *et al.*, 1991b, Dorigo, *et al*., 1996), *Ant-Q* (Gambardella and Dorigo, 1995; Dorigo and Gambardella, 1996)*, Ant Colony System* (Dorigo and Gambardella, 1997a; Dorigo and Gambardella, 1997b), *MAX-MIN AS* (Stutzle and Hoos, 1996; Stutzle and Hoos, 2000; Stutzle 1999), *Rank-Based AS* (Bullnheimer, *et al.*, 1997)

In this research work, we propose and apply ACO to arrange a number of machines in a production line so that to minimize the total amount of backward flows. This is a first approach towards the development of a method to configure more complex and more general layouts of machines in the shop floor based on the ACO meta-heuristic algorithm.

2. LINEAR MACHINE LAYOUT PROBLEM

For Flexible Manufacturing Systems (FMS) the layout design is even more crucial than in conventional manufacturing. Whereas a variety of methods that implement complex networks and layouts are available, the linear or single-row layout is the most commonly implemented layout in manufacturing systems due to its simplicity. There are many shapes of linear layouts, such as straight line, circular loop, U-shape and serpentine line (Haragu and Kusiak, 1998) (Figure 1).

The configuration of the production line is heavily dependent on the material-handling system. Apart from its configuration, the production line is characterized by the flow of material as unidirectional or bidirectional. In the latter, four different types of flow movement can occur: Repeated operations, in-sequence operations, bypassing-operations and backtracking operations (Figure 2). The most desirable is the in-sequence operation due to its unidirectional movement.

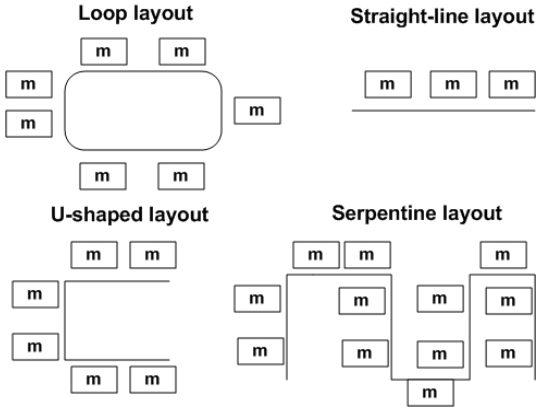

Fig.1. Alternative configurations of single-row layouts

Backward flow is the least desirable since it costs and complicates the flow more than the forenamed flow movements.

The ideal scenario would include only in-sequence moves. In practice, however, bypassing and backtracking of jobs as they pass down the line is inevitable. The designer of a single-row layout has to find the optimal arrangement of machines for such a production line. The optimality depends on the criteria and the restrictions that are posed. There are four criteria, which a designer could take into account (Ponnambalam and Ramkumar, 2001): Minimization of the number of backtracking movements, minimization of total backtracking flow distance, maximization of in-sequence movements and minimization of the flow distance.

Carrie (1975) was the first to study the linear layout problem and several approaches have been proposed since then (Aneke and Carriw, 1986; Lee, 1991; Kouvelis and Chiang, 1992; Sarker *et al.*, 1994; You-Dong, 1997; Ponnambalam and Ramkumar, 2001).

In this paper, we investigate the solution to the "minimal backward-flow" model, (minimization of total backtracking flow distance) introducing the ACO algorithm for a simplified model.

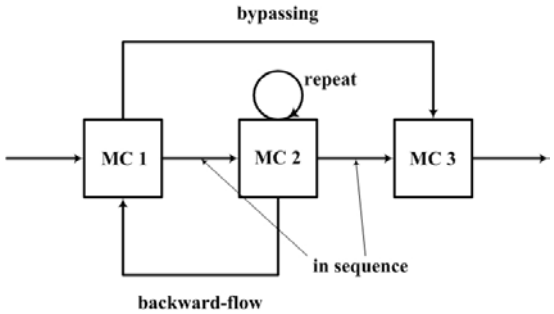

Fig.2. The Four different flow movements in a linear layout

2.1 Minimal backward-flow model.

This model is developed by trying to minimize the total amount of backward flows for a production cell, as it is indicated but its name. For this model, we make the following assumptions:

- Only one machine of each type is allowed in the line (no duplications of machines are allowed).
- The cost of material flows is proportional to the number of parts and the distance of flows.
- Each machine is regarded as a point and the distance between machines is "1", the unit distance.

The distance between the initial input point of parts and the first machine is also regarded as the unit distance. In Figure 3 the adopted conventions are depicted.

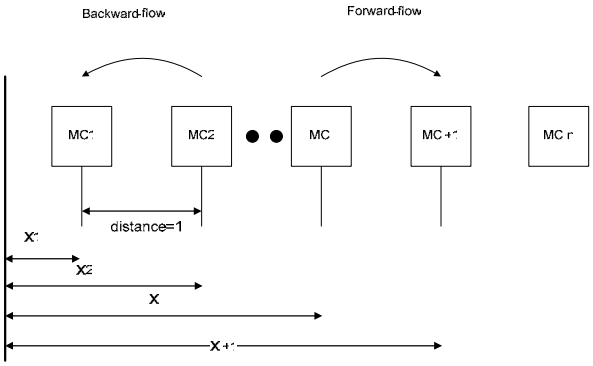

Fig. 3. Linear machine layout and corresponding notation

It must be mentioned that minimization of backtracking does not necessarily reduce the bi-directional travel distance. Such a requirement would probably lead to a different formulation.

Therefore, the problem can be described as follows:

Having *n* machines and *m* items of parts to be produced and for each item a corresponding demand of d_j (*j = 1, 2. . . m),* place the machines in such an order so that to minimize the backward flows. The quantity that has to be minimized is the total number of backtracking steps.

Thus, for this problem, following the notation of Sarker, *et al*., (1994), we seek to minimize:

$$TB = \sum_{j=1}^{M}\sum_{i=1}^{M} r_{ij} b_{ij} = \mathbf{R} \cdot \mathbf{B} \qquad (1)$$

where M is the number of machines, $\mathbf{R} = \left[r_{ij} \right]_{M \times M}$ is the requirement matrix, $\mathbf{B} = \left[b_{ij} \right]_{M \times M}$ is the backtrack matrix, r_{ij} is the number of total moves from machine i immediately to machine j and b_{ij} is the number of backtrack steps from machine i to machine j. For a more rigorous analysis the interested reader is referred to (Sarker, *et al*., 1994).

Example 1

For example, suppose that we have 3 machines and we want to produce 2 items (10 parts of the 1st item and 15 parts of the 2nd item). In order to produce those parts, we have to use the 3 machines in the following order:

Item 1 1-2-3-2-3-1 (10 parts)
Item 2 3-2-1-3-2-3-1-2 (15 parts)
and the machine layout is 1-2-3.

In order to calculate the total cost, we construct the matrices **R**, **B** and we multiply them, element by element and we sum them according to equation 1.

$$\mathbf{B} = \begin{array}{c} \\ From \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} To & & \\ 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 2 & 1 & 0 \end{bmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}$$

$$\mathbf{R} = \begin{array}{c} \\ From \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} To & & \\ 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 25 & 15 \\ 15 & 0 & 35 \\ 25 & 40 & 0 \end{bmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}$$

Therefore the total backtracking cost is found: $TB = 1 \times 15 + 2 \times 20 + 1 \times 40 = 105$. As it is obvious the requirement matrix remains constant, whereas the backtrack matrix changes according to the arrangement of the machines. This function is both complicated and difficult to estimate before all the machines are in place.

In the proposed approach starting from an initial machine, we attempt to determine the machine that has to follow so that the aforementioned cost function is minimized, using local information. The construction of new solutions is based on an ACO algorithm incorporating the information contained in the requirement matrix.

3. ACO IMPLEMENTATION

The employment of the ACO algorithm for this particular problem is performed by considering that the artificial ants are moving from one machine to another building a path, which is a candidate solution for this problem. That is, starting from a machine, the ant proceeds by selecting the machine to be put next in the layout. The selection of the next machine has to be done based on a trade-off between the pheromone (τ), which is deposited on the arcs connecting the current machine with the other machines and a heuristic (n) function, which measures locally the quality of the machine that can be added to the current partial solution.

Adopting the Ant Colony System (ACS) approach (Dorigo and Gambardella, 1997a; Dorigo and

Gambardella, 1997b) we construct a candidate solution using the following "tour" formulation.

3.1 "Tour" construction

When the *k*-th ant is located at machine *i*, it chooses to move to machine *j*, according to the so-called *pseurorandom proportional rule,* given by:

$$j = \begin{cases} \max_{u \notin J_i^k} \left\{ [\tau_{iu}(t)]^{\alpha} \cdot [\eta_{iu}(t)]^{\beta} \right\}, if \quad q \le q_0 \\ J \qquad\qquad\qquad , \quad if \quad q > q_0 \end{cases} \tag{2}$$

where q_0 is a parameter that it is selected by the user in the interval [0,1], and *J* is a machine that has not been used so far and is selected with probability:

$$p_{iJ}^k(t) = \frac{[\tau_{iJ}(t)]^{\alpha} \cdot [\eta_{iJ}(t)]^{\beta}}{\sum_{l \notin J_i^k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}(t)]^{\beta}}, \quad if \quad J \notin J_i^k \tag{3}$$

among the other candidates. J_i^k denotes the tabu list (forbidden list), which contains all the machines that have been so far used including the current machine i, η_{ij} is a heuristic value that is available *a priori*, τ_{ij} is the pheromone trail, α and β are two parameters, which determine the relative influence of the pheromone trail and the heuristic information. In other words, with probability q_0 the ant makes the best possible move as indicated by the learned pheromone trails and the heuristic information (in this case, the ant is exploiting the learned knowledge), while with probability (1- q_0) it performs a biased exploration of the arcs. Tuning the parameter q_0, it allows modulation of the degree of exploration and the choice of whether to concentrate the search of the system around the best-so-far solution or to explore other tours. As it is obvious the choice depends heavily on the quantities $\eta_{ij}(t)$ and $\tau_{ij}(t)$.

3.2 *Pheromone*

Real ants follow pheromone trails. Artificial ants tend to follow this artificial-pheromone. The quantity $\tau_{ij}(t)$ corresponds to the "arc" connecting machine *i* to machine *j*. Or to be more accurate corresponds to the directional arc from *i* to *j*. This means that usually it is: $\tau_{ij}(t) \neq \tau_{ji}(t)$. This quantity changes after the completion of any search for all ants. At the first step of the algorithm all pheromone trails are initiated to a value τ_0. This initial value τ_0 can either be set to a small number or preferably, as in the case of the Traveling Salesman Problem (TSP), to the value of $1/nC^{nn}$, which has been found to be a good choice, where n is the number of cities (number of machines for our problem) and C^{nn} is the length (cost) of the solution found using the nearest neighbour heuristic.

3.3 Heuristic information (Visibility)

The heuristic function (which some times is referred to with the term visibility, reminding its origin from the TSP problem) is a quantity, which in our problem (a static problem) doesn't change over iterations ("iteration-invariant"), so a more proper notation is: $\eta_{ij} = 1/d_{ij}$.

Where d_{ij} in the original implementation of the TSP is simply denoting the distance between town *i* and *j*. In our case, d_{ij} implements a "local" cost in accordance with the total cost function that has to be minimized. This means that d_{ij} measures the immediate cost that we have to pay by placing machine *j* after machine *i*. This is calculated by summing all the backflows created by this arrangement within a unity distance, i.e. by restricting our search to only adjacent steps in the production phase. In other words, d_{ij} is equal to the element r_{ji} of the requirement matrix.

3.4 Pheromone Trail Update

Global update. Only one ant (the best-so-far ant) is allowed to add pheromone after each iteration. Thus, the update in ACS is implemented by the following equation:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho \Delta \tau_{ij}^{bs}, \quad \forall (i,j) \in T^{bs} \tag{4}$$

where T^{bs} is he best-so-far tour (machinery layout), $\Delta \tau_{ij}^{bs} = 1/C^{bs}$, where C^{bs} is the "length", i.e. the calculated cost for this particular arrangement of machines T^{bs} and the parameter ρ represents pheromone evaporation (with $0 \le \rho \le 1$). For ACS algorithm, during the pheromone trail update, both evaporation and new pheromone deposit, it only applies to the arcs of the best constructed solution so far.

Local update. The ants use a local pheromone update rule, which is applied immediately after having crossed an arc (*i, j*) during the tour construction:

$$\tau_{ij} = (1-\xi)\tau_{ij} + \xi\tau_0 \tag{5}$$

Where ξ is a parameter ($0<\xi<1$) and τ_0 is the initial value for the pheromone trails. The effect of the local updating rule is the following: every time an ant uses an arc (*i, j*), its pheromone trail τ_{ij} is reduced, so that this arc becomes less desirable for the following ants. This approach allows an increase in the exploration of arcs, which have not been visited and, in practice, this algorithm does not show a stagnation behavior (ants do not converge to the generation of a common path).

4. RESULTS

In order to test the proposed method we examine an FMS problem, where all the involved quantities are

generated randomly and are summarized in the following table 1.

For this particular problem the optimal arrangement of the machines is: 7, 8, 3, 2, 6, 9, 5, 1, 4 with a total number of backtracking steps equal to 2923 (total cost). The total number of possible solutions is 9!= 362880. On the other hand, the worst case scenario would be to arrange the machines in the following order 2, 1, 4, 7, 5, 6, 9, 8, 3 with a total cost of 4980.

Table 1: 8 Processes, with their corresponding routes and demands

Process #	Route information	# of parts
1	[1 6 8 9 3 5 7 4 3 8 6 8 2 3]	8
2	[2 6 1 8 9 5 2 1 6 2 9 5 6 8 4 3 4]	22
3	[8 7 3 4 1 8 5 6 2 3 1]	33
4	[3 4 6 9 2 1 7 2 8 1]	12
5	[8 7 3 1 4 1 5 6 2 9 3 1]	14
6	[9 8 7 2 3 4 5 1 5 7 6 2 3 1]	23
7	[3 7 9 4 9 2 5 1 7 8 2 8 6 3 2]	39
8	[3 7 2 4 6 2 9 1 9 5 8 3 4]	28

Our "colony" consists of 9 ants (equal to the number of the machines) and in each experiment the algorithm is let to run for 1000 iterations. For each execution of the experiment, we have a total of 9000 constructed solutions (obviously some of them are repeated more than one times), which is only 2.5% of the total number of candidate solutions. Since the algorithm is stochastic in nature, we repeat the experiments 10 times and calculate the average performance. In Figure 3 we depict the evolution of the best solution for each one of the 10 trials along with their average.

The average cost achieved is 2932 and in addition to that, the algorithm 4 times out of 10 also finds the global optimal solution (2923).

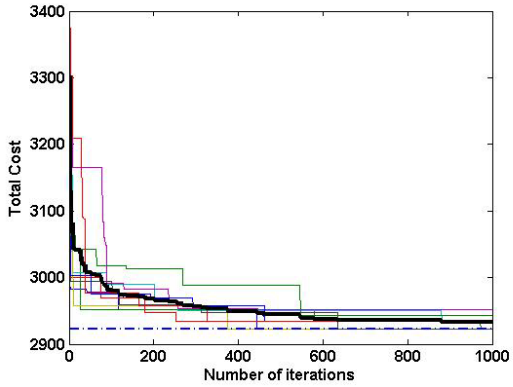

Fig. 3. The evolutions of C^{bs} for 10 different runs of the experiment. The thick line corresponds to the average of those 10 trials, while the dashed line on the bottom marks the global best value.

5. CONCLUSIONS

In this work, we propose and adapt the ACS paradigm for the solution of a simplified linear layout problem. The results are encouraging indicating that this method has potentials and with further investigation could be used for more complicated problems concerning machinery layout for shop floors. It is proven that by exploring only a small number of candidate solutions, we are able to find a good (near-optimal) solution even though without optimising the parameter settings of the algorithm. In future work, we are planning to use an evolutionary approach (such as Particle Swarm Optimization) for the tuning of the set of parameters.

We must also point out that this particular problem is slightly different compared to the traditional TSP problem. In the TSP problem the relative positions of the node numbers are more important than the absolute positioning of the node numbers. However, in our case this is not true, since the choice of the first positioned machine is also important. However, no special action is taken in favour of the first machine in the layout and the machines are uniformly located in the first place. The next step in our research will be to develop an ant-like procedure to account for the first machine in the layout, favoring the one that tends to give better results.

REFERENCES

Aneke, N.A.G., and A.S. Carrie (1986). A design technique for the layout of Multi-product flow lines. *International Journal of Production Research*, **vol 24**, pp. 471-481.

Bullnheimer, B., R.F. Hartl and C. Strauss (1997). A new ranked-based version of the Ant System: A computational study. *Central European Journal for Operations Research and Economics*, **vol 7**, **no 1**, pp. 25-38.

Carrie, A.S. (1975). Layout of Multi-product lines. *International Journal of Production Research*, **vol 13**, pp. 541-575.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms* [in Italian]. PhD thesis, Dipartimento di Electronica, Politecnico di Milano, Milan.

Dorigo, M., and L.M. Gambardella (1996). A study of some properties of Ant-Q. In H. Voigt, W. Ebeling, I. Rechenberg, & H. Schwefel (Eds.), *Proceeding of PPSN-IV, Fourth International Conference on Parallel Problem Solving from Nature*, vol 1141 of Lecture Notes in Computer Science (pp.656-665). Berlin, Springer-Verlag.

Dorigo, M. and L. M. Gambardella (1997a). Ant colonies for the traveling salesman problem. *Biosystems*, **vol 43, no 2**, pp. 73-81.

Dorigo, M. and L.M. Gambardella (1997b). Ant Colony System: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, **vol 1**, **no 1**, pp. 53-66.

Dorigo, M., G. Caro and L. M. Gambardella (1999). Ant Algorithms for Discrete Optimization. *Artificial life*, **vol 5**, pp. 137-172.

Dorigo, M., V. Maniezzo and A. Colorni (1991a). Positive feedback as a search strategy. *Technical report 91-016*, Dipartimento di Electronica, Politecnico di Milano, Milan.

Dorigo, M., V. Maniezzo and A. Colorni (1991b). The Ant System: An autocatalytic optimizing process. *Technical report 91-016* revised, Dipartimento di Electronica, Politecnico di Milano, Milan.

Dorigo, M., V. Maniezzo and A. Colorni (1996). Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, **vol 26**, **no 1**, pp. 29-41

Dorigo M. and T. Stötzle (2004). *Ant Colony Optimization*, A Bradford Book, The MIT Press, Massachusetts Institute of Technology.

Gambardella, L.M. and M. Dorigo (1995). Ant-Q: A reinforcement learning approach to the travelling salesman problem. In A. Prieditis and S. Russell (Eds), *Proceedings of the twelfth International Conference On Machine Learning* (ML-95) (pp. 252-260). Palo Alto, CA, Morgan Kaufmann.

Heragu, S.S. and A. Kusiak (1998). Machine layout problem in flexible manufacturing systems. *Operations Research*, **vol 36**, **no 2**, pp. 258-268

Kouvelis, P. and W. C. Chiang (1992). A simulated annealing procedure for single row layout problems in flexible manufacturing systems. *Int. J. Prod. Res*. **vol 30**, **no 4,** pp 717-732.

Lee, C.E.C. (1991). An integrated methodology for the analysis and design of cellular flexible assembly systems. Ph.D. thesis Purdue University West Lafayette, IN, USA.

Ponnambalam, S.G. and V. Ramkumar (2001). A genetic Algorithm for the Design of a Single-Row Layout in Automated Manufacturing Systems. *Int J. Adv. Manuf. Technol.,* **vol 18**, pp. 512-519.

Rich, E. and K. Knight (1991). *Artificial Intelligence*, Mcgraw-Hill, Inc, International Edition.

Sarker, B.R., W.E. Wilhelm and G. L. Hogg (1994). Backtracking and its Amoebic Properties in One-dimensional Machine Location Problems. *J. Opl Res. Soc.,* **vol 45**, **no 9**, pp 1024-1039.

Stutzle, T. (1999). Local search Algorithms for Combinatorial Problems: Analysis, Improvements and New Applications. vol. 220 of DISKI. Sankt Augustin, Germany, Infix

Stutzle, T. and H.H. Hoos (1996). Improving the Ant System: A detailed report on the MAX-MIN Ant System. *Technical Report AIDA-96-12*, FG Intellektik, FB Informatik, TU Darmstadt, Germany.

Stutzle, T. and H.H. Hoos (2000). MAX-MIN Ant System. *Future Generation Computer Systems*, **vol 16**, **no 8**, pp. 889-914.

You-Dong, W. (1997). A linear programming approach to linear machine layout problem. *The Journal of Industrial Mathematics Society*, **vol 47**, **no 2**, pp. 59-69.