A COMPARISON OF HEURISTICS FOR MEAN FLOW TIME OPEN SHOP SCHEDULING

Heidemarie Bräsel * **Andre Herms** * **Marc Mörig** *
Thomas Tautenhahn * **Jan Tusch** * **Frank Werner** *
Per Willenius *

* *Faculty of Mathematics, Otto-von-Guericke University, Postfach 4120, D-39016, Magdeburg, Germany, e-mail: heidemarie.braesel@mathematik.uni-magdeburg.de, frank.werner@mathematik.uni-magdeburg.de*

Abstract: In this paper, the problem of scheduling n jobs on m machines in an open shop environment is considered so that mean flow time becomes minimal. Since this problem is strongly NP-hard, different constructive and iterative heuristic algorithms are developed and discussed. Computational results are presented for problems with up to 50 jobs and 50 machines, respectively. The quality of the solutions is estimated by a lower bound for the corresponding preemptive open shop problem and by an alternative estimation of mean flow time.

Keywords: Schedule algorithms, Multimachine, Optimization, Implementation

1. INTRODUCTION

In an open shop scheduling problem, a set of n jobs $J_1, J_2, \ldots, J_n$ has to be processed on a set of m machines $M_1, M_2, \ldots, M_m$. The processing of a job J_i on machine M_j is denoted as operation (i, j), and the sequence in which the operations of a job are processed on the machines is immaterial. It is assumed that the processing time t_{ij} of each operation (i, j) is given. As usual, each machine can process at most one job at a time and each job can be processed on at most one machine at a time.

Open shop scheduling problems arise in several industrial situations. For example, consider a large aircraft garage with specialized work-centers. An airplane may require repairs on its engine and electrical circuit system. These two tasks may be carried out in any order but it is not possible to do these tasks on the same plane simultaneously. Other applications of open shop scheduling problems in automobile repair, quality control centers, semiconductor manufacturing, teacher-class assignments, examination scheduling, and satellite communications are described in (Kubiak *et al.*, 1991; Liu and Bulfin, 1987; Prins, 1994).

Most papers in the literature dealt with the minimization of makespan. Gonzalez and Sahni (1976) presented an $O(n)$ algorithm for the two-machine open shop problem denoted as $O2||C_{max}$. The preemptive problem can be solved in polynomial time for an arbitrary number of machines, see (Gonzalez and Sahni, 1976). However, slight generalizations of the two-machine non-preemptive problem lead already to NP-hard problems. In view of the NP-hardness of problem $O||C_{max}$, heuristic and branch and bound algorithms have been developed for this problem. Bräsel *et al.* (1993) developed several constructive heuristics based on matching algorithms (which determine subsets of operations that can run simultaneously) as well as on the insertion of operations into par-

tial schedules combined with beam search. Other heuristic algorithms for the open shop problem with minimizing makespan have been given e.g. in (Alcaide *et al.*, 1997; Liaw, 2000). Exact algorithms for the open shop problem with minimizing makespan have been given e.g. in (Brucker *et al.*, 1997; Gueret *et al.*, 2000; Dorndorf *et al.*, 2001). The latter algorithm is able to solve all but one problems of a benchmark set with 15 jobs and 15 machines in about 12 minutes on the used computer.

There exist only a few papers dealing with the minimization of mean flow time although from a practical point of view, often the minimization of the latter criterion is more important. In (Achugbue and Chin, 1982), it has been proved that the two-machine open shop problem with minimizing the sum of completion times (also denoted as mean or total flow time minimization) is NP-hard in the strong sense. In (Liaw *et al.*, 2002), the problem of minimizing total completion time with a given sequence of jobs on one machine has been considered (also denoted as $O|GS(1)|\sum C_i$). This problem is already NP-hard in the strong sense even in the case of two machines. First, a lower bound has been derived based on the optimal solution of a relaxed problem in which the operations on every machine may overlap except for the machine with a given sequence of jobs. Then a branch and bound algorithm has been presented and tested on square problems with $n = m$. The algorithm was able to solve all problems with 6 jobs in 15 minutes on average and most problems with 7 jobs within a time limit of 50 hours with an average computation time of about 15 hours for the solved problems. A heuristic algorithm has also been proposed which is an iterative dispatching procedure. Concerning approximation algorithms with performance guarantee, the currently best result has been given in (Queyranne and Sviridenko, 2002).

In this paper, the non-preemptive open shop scheduling problem with mean flow time minimization is considered. A comparison of different types of constructive and heuristic algorithms is performed in order to point out the differences to the makespan minimization case. In contrast to all other papers, which consider only square problems with $n = m$, we also investigate the influence of the ratio of n and m on the choice of an appropriate algorithm. The remainder of the paper is organized as follows. In Section 2, we introduce some basic notions and present estimates for the optimal objective function value. In Section 3 several constructive algorithms are described, and in Section 4 the iterative algorithms applied in our tests are sketched. Some computational results are briefly discussed in Section 5.

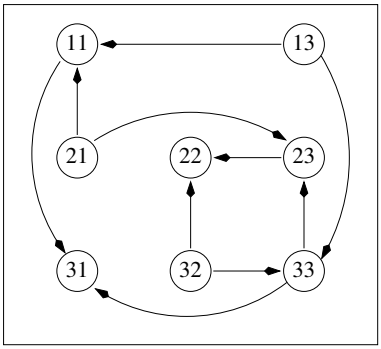

Fig. 1: The sequence graph

2. BASIC NOTIONS AND SCHEDULE EVALUATION

In most papers on open shop problems, a feasible schedule is represented by a permutation of all operations. In the following, we use the digraph $G(MO, JO)$ with operations as vertices and arcs between two immediately succeeding operations of a job or on a machine. If we place the operations of job J_i in the i-th row and the operations on machine M_j into the j-th column, then $G(MO, JO) = G(MO) \cup G(JO)$ where $G(MO)$ contains only horizontal arcs (describing the machine order of the jobs) and $G(JO)$ contains only vertical arcs (describing the job orders on the machines). A combination of machine orders and job orders (MO, JO) is feasible, if $G(MO, JO)$ is acyclic. An acyclic digraph $G(MO, JO)$ is called *sequence graph*. In this case all above graphs represent partial orders on the set of operations. For a small example with $n = 3$ jobs and $m = 3$ machines, a sequence graph $G(MO, JO)$ is given in Fig. 1.

Similarly as in (Bräsel *et al.*, 1993) and (Werner and Winkler, 1995), we describe a sequence graph $G(MO, JO)$ by its *rank matrix* $A = (a_{ij})$, i.e., the entry $a_{ij} = l$ means that a path to operation (i, j) with maximal number of operations has l operations. Notice that for each $a_{ij} = k > 1$, integer $k - 1$ occurs as entry in row i or column j (or both). Now we assign the processing time t_{ij} of operation (i, j) as the weight of this operation in $G(MO, JO)$. The computation of a longest path to the vertex (i, j) with (i, j) included in an acyclic digraph $G(MO, JO)$ yields the completion time c_{ij} of operation (i, j) in the semiactive schedule $C = (c_{ij})$. Moreover, C_i denotes the completion time of job J_i, and r_{ij} denotes the head of operation (i, j). The use of sequence graphs resp. rank matrices avoids the disadvantage of using a permutation of the operations, where several permutations represent the same solution.

In (Bräsel and Hennes, 2004), the above model has been generalized to the preemptive case and a lower bound for the open shop problem $O|pmtn|\sum C_i$ has been derived which can also be taken for the non-preemptive problem. This lower bound is as follows. Let

$$T_i = \sum_{j=1}^{m} t_{ij} \qquad \text{and} \qquad \overline{T}_j = \sum_{i=1}^{n} t_{ij}$$

and suppose that jobs and machines are ordered such that

$$T_1 \leq T_2 \leq \ldots \leq T_n$$

and

$$\overline{T}_1 \leq \overline{T}_2 \leq \ldots \leq \overline{T}_m.$$

Obviously, we have $C_i \geq T_i$ in a feasible schedule and any unavoidable waiting time of job J_i increases the completion time C_i. The goal is to identify such unavoidable waiting times of jobs. We can assume that $n = m$ (otherwise we introduce dummy jobs or machines). First, let $T_1 > \overline{T}_1$, i.e., the total processing time of job J_1 is greater than the machine load on M_1 and let $h_1 := T_1 - \overline{T}_1$. Considering the intervals in which job J_1 is processed, there are subintervals of total length h_1 with the following property: at each time inside these subintervals, there can be at most $n-2$ jobs processed simultaneously with J_1 since one machine different from M_1 is occupied by job J_1 and machine M_1 is idle. Thus, one (several) job(s) with

$$T_j < \overline{T}_j \tag{1}$$

must wait until value T_j has been increased to $\overline{T}_j$. Therefore, one chooses the longest job J_k with $T_k < \overline{T}_k$ and performs the following computations:

$$H := \min\{T_k + h_1, \overline{T}_k\}; \quad h_1 := h_1 - H + T_j;$$
$$T_j := H.$$

This procedure is repeated with other jobs satisfying inequality (1) until $h_1 = 0$. If $T_1 < \overline{T}_1$, then there is an unavoidable idle time on machine M_1, and the rest is symmetric to the first case. Then jobs $J_2, J_3, \ldots, J_n$ are considered in a similar way and finally, the lower bound for the preemptive problem is equal to sum of the final T_i values, see also (Bräsel and Hennes, 2004). We also use an alternative estimate for the optimal function value which is based on the polynomial solution of a modified open shop problem with unit processing times.

3. CONSTRUCTIVE ALGORITHMS

Among constructive algorithms which generate one or a small number of solutions (without changing settled decisions), we consider matching algorithms, the generation of active and nondelay schedules, insertion and appending procedures combined with beam search.

(a) Matching algorithms

The first type of algorithms is based on matching procedures. They have been suggested in (Bräsel *et al.*, 1993) for the makespan minimization problem and generate so-called *rank-minimal schedules*. Without loss of generality, we assume that we have a square problem with $n = m$ (otherwise we introduce dummy jobs or dummy machines such that $n = m$). The algorithm successively determines n operations having rank 1 in the graph $G(MO, JO)$, n operations having rank 2, and so on. This is done by solving weighted bipartite maximum cardinality matching problems.

(b) Generation of active and nondelay schedules

A schedule is called *active* if no operation can be started earlier without delaying some other operation. A schedule is called *nondelay* if no machine is left idle provided that it is possible to process some job. The algorithms for constructing active and nondelay schedules repeatedly append operations to a partial schedule. Starting with an empty schedule, operations are appended as follows:

- To construct a nondelay schedule, we determine the minimal head r of all unscheduled operations. At time r, there exist both a free machine and an available job. To maintain the nondelay property of the schedule, we have to append an operation which can start at time r. Among all operations (i, j) with $r_{ij} = r$, choose one according to some priority dispatching rule.
- To construct an active schedule, we determine the minimal possible completion time EC of all unscheduled operations with respect to their heads and processing times. The next operation to append is chosen among all unscheduled operations which can start before EC. Again, that choice is made by a priority dispatching rule.

In our tests, we have used the following priority dispatching rules: $FCFS$ (first come first served); ECT (earliest completion time); SPT (shortest processing time); LPT (longest processing time).

(c) Insertion algorithms combined with beam search

Insertion algorithms combined with beam search have been suggested in connection with shop scheduling problems, e.g. in (Bräsel *et al.*, 1993) for the open shop problem and in (Werner and

Winkler, 1995) for the job shop problem. They are restricted branch and bound procedures which are based on an enumerative algorithm for schedules. The insertion algorithm works as follows. According to some chosen insertion order, operations are successively inserted into partial sequence graphs. When a new operation is inserted, all precedence constraints previously established are maintained. Let v_k operations of job J_k and u_l operations on machine M_l be already sequenced. Then there exist $(v_k+1)\cdot(u_l+1)$ possibilities to insert operation (k,l), namely operation (k,l) can be inserted on the i-th position in the machine order of job J_k, $1 \leq i \leq v_k+1$ and on the j-th position in the job order on machine M_l, $1 \leq j \leq u_l+1$ (however, not every combination of positions leads to a sequence graph). In all cases, the ranks of all successors of the inserted operation have to be updated.

Algorithm Beam-Insert is determined by fixing the insertion order IO, the estimation LB for the objective function value of a partial schedule, the beam width k and the son selection procedure SEL, i.e., the partial sequence graphs that are selected from the set of generated graphs for further considerations. Typically, the insertion order IO is determined by some well-known priority rule (see generation of active and nondelay schedules). In our tests, we have considered the rules $RANDOM, SPT$, LPT and $FCFS$. A partial schedule usually estimated by means of a rough lower bound for the objective function value of an arbitrary completion of the current partial sequence graph and schedule, respectively. We use the following bound:

LBP: For a partial schedule, the value $\sum C_i$ is estimated by the sum of the completion times of the currently last scheduled operation of each job.

The beam width k denotes the number of paths that are considered in the branching tree. To select k partial sequence graphs, we have considered the following two variants for the son selection procedure SEL:

SEL1: Select in each step the k best sons from the whole set of partial sequence graphs, i.e., the selected sons do not necessarily have different fathers.

SEL2: For each of the k fathers select the best son, i.e., all sons have different fathers (as long as we do not have k fathers the first criterion is applied).

(d) Appending procedures combined with beam search

In some initial tests we have found that for certain types of problems, even nondelay schedules produced better results than the Beam-Insert algorithm. This leads to the idea to combine the generation of nondelay schedules with beam search. Next, we discuss deterministic settings of the parameters of this procedure. In each step, a partial sequence graph is extended by appending some operation. For selecting this operation, the smallest possible head of some unscheduled operation is determined. In the case when this operation is not uniquely determined (which is in particular the case at the beginning of the procedure), we apply some tie-breaking rule TBR to select an operation (i,j). We take as tie-breaking rule a specific priority dispatching rule. In our tests, we used $TBR \in \{RANDOM, SPT, LPT, FCFS\}$.

In order to generate several successors from a partial sequence graph, we considered two son generation procedures SG for generating the successors of the current partial sequence graph(s), namely a machine-oriented ($SG = MO$) and a job-oriented ($SG = JO$) appending of the next operation. This means that, considering the selected operation (i,j), either an unscheduled operation on machine M_j or an unscheduled operation of job J_i is sequenced such that a nondelay schedule results.

Algorithm Beam-Append is characterized by the tie-breaking rule TBR, the son generation procedure SG, the estimation LB of the partial schedule, the beam width k, and the son selection type SEL. Parameters k and SEL are similar to the corresponding parameters for procedure Beam-Insert. For the evaluation of the generated sons, we have considered in addition to LBP the following estimation for the objective function value:

LBC: For the Beam-Append procedure, it is known that all unscheduled operations of a job must be sequenced later than the currently last scheduled operation of this job. Therefore, the completion time C_i of job J_i is estimated by the largest completion time of the scheduled operations of this job plus the sum of the processing times of the unscheduled operations of job J_i.

4. ITERATIVE ALGORITHMS

Among the iterative algorithms, simulated annealing, tabu search, a genetic algorithm and an ant colony algorithm have been considered. While standard metaheuristics such as the first two ones are single trajectory methods (i.e. they manipulate a single feasible solution), the latter two algorithms work with a population. For the comparative study, the stopping criteria have been settled such that all iterative algorithms consume a similar amount of computational time.

(a) Single trajectory methods

Simulated annealing sometimes accepts moves that lead to a neighbor with worse objective func-

tion value. In particular, if a neighbor (rank matrix) A' of the current starting solution A with better function value has been generated, it is always accepted. Otherwise, the acceptance probability of a non-improving move depends on the difference Δ in the objective function values of A' and A and the temperature control parameter T and is equal to $\exp(-\Delta/T)$. Initially, the temperature T is high and then it decreases to a value close to zero as the search proceeds. A simulated annealing algorithm mainly depends on the chosen neighborhood and the cooling scheme for the temperature.

First, we briefly discuss the generation of neighbors of a current solution described by a sequence graph $G(MO, JO)$ resp. rank matrix A. In the case of a job shop problem, often a neighbor is generated by interchanging two adjacent jobs in exactly one machine order. We denote this neighborhood as machine oriented API-neighborhood, abbreviated as $API(MO)$. In an open shop problem we can, due to symmetry, consider a neighborhood based on adjacent pairwise interchanges in the machine order of the jobs, abbreviated as $API(JO)$. In our algorithms, we use the union of both neighborhoods, abbreviated as $API(MO + JO)$. A second neighborhood considered is $crit - API(MO+JO)$, which is a restricted $API(MO+JO)$ neighborhood in which a neighbor must satisfy a necessary condition for an improvement of the makespan value. This neighborhood is based on the so-called block approach originally introduced for shop scheduling problems with makespan minimization. Moreover, we consider the neighborhood $k - API(MO + JO)$, in which a neighbor is generated from the current sequence graph $G(MO, JO)$ by generating consecutively up to k neighbors in the $API(MO + JO)$ neighborhood (i.e. a path containing up to k arcs in the resulting neighborhood graph is generated). Finally, we consider the h-reinsertion neighborhood. In the latter case, $g \in \{1, 2, \ldots, h\}$ chosen operations of the same job are deleted and then successively reinserted at the best position, i.e. the insertion algorithm is applied to the operations that have been deleted. As the cooling scheme, a geometric, a Lundy-Mees and a linear reduction scheme have been considered.

Tabu search is an iterative procedure that moves in each iteration to the best neighbor investigated which has not necessarily a better objective function value. To avoid cycling and to escape from a poor local optimum, a tabu restriction is used that makes selected attributes of these moves forbidden (tabu). Tabu restriction is enforced by a tabu list L which stores the move attributes to avoid reversals of moves. The tabu list contains the moves describing the last l solutions visited, where l is the size of the tabu list. It controls the memory of the search process (we use a short-term memory in our implementation of tabu search). In the experiments, we mainly test the influence of the neighborhood, the size of the tabu list and the number of generated nontabu neighbors in one iteration.

(b) Population based methods

Genetic algorithms are general search techniques based on natural selection mechanisms and genetics. They manipulate and maintain a population of schedules represented in some encoding scheme (chromosomes). We choose the rank matrix instead of the common linear order representation of the operations. The chromosomes are modified in order to produce offspring by application of genetic operators (mutation and crossover). A mutation changes the rank of one operation in the rank matrix, and then all ranks are updated so that the relative order of all remaining operations is maintained. The crossover operator recombines the structure of two chromosomes into one (or two) new chromosome(s) distinct from the originals. Two rank matrices are selected, some operations of a set H are chosen and the ranks or these operations in both rank matrices are interchanged. Then both chromosomes are completed by maintaining the relative order of the remaining operations not contained in H. After having generated all offspring we apply fitness proportional selection on the union of the old and new generations to obtain a new population. In addition, the elitist strategy has been considered, i.e. the rank matrix from some population is directly inserted into the next generation. We mainly investigate the influence of the probabilities for applying the genetic operators and the population size. Moreover, **hybrid genetic algorithms** have also been tested, where to any generated offspring a fast local search procedure is applied before selection.

The basic idea of an **ant colony algorithm** comes from the ability of ants to find shortest paths from their nest to food locations. In a combinatorial problem, the ant iteratively builds a solution of the problem. This procedure is conducted using at each step a probability distribution which corresponds to the pheromone trail in real ants. Once a solution is completed, pheromone trails are updated according to the quality of the solution constructed (i.e. cooperation between ants is performed by the common structure which is the shared pheromone matrix). In our implementation, we follow the strategy given e.g. in (Blum, 2005), which turned out to work particularly good for the open shop makespan minimization problem. At each iteration, a number of ants probabilistically construct solutions. In particular, by means of a randomized Beam-Append procedure, rank matrices representing nondelay

and active schedules are constructed. Then an iterative improvement procedure is applied to improve the constructed solutions. Finally, some of the constructed solutions are used for performing an update of the pheromone values which aim at increasing the probability to generate high quality solutions. The pheromone values encode for any two operations belonging to the same job or being processed on the same machine the desirability of performing a particular operation before the other one. In our experiments, we test in particular the parameters of the probabilistic beam search procedure.

5. COMPUTATIONAL RESULTS

For the comparative study, we have considered all pairs (n, m), $n \neq m$, with $n \in \{10, 20, 30, 40, 50\}$ and $m \in \{10, 20, 30, 40, 50\}$. Additionally, we have considered square problems with $n = m \in \{15, 25, 35, 45\}$. For each combination (n, m) we generated 50 instances with processing times from the interval [1,20] and 50 instances with processing times from the interval [1,100]. From our detailed computational experiences, we sketch here only briefly some aspects of the results (a detailed comparison of the results with the different types of algorithms in dependence on the problem type is given in the talk).

- For the open shop problem with mean flow time minimization, the choice of an appropriate constructive solution procedure strongly depends on the relationship between the number n of jobs and the number m of machines. Matching procedures do not work well. The generation of nondelay schedules is superior to the generation of active schedules.
- For problems with $n/m \leq 2/3$, Beam-Insert is an excellent constructive algorithm. For many instances, even the lower bound is met for some variant, and the average percentage deviations are small in this case.
- For problems with $n \geq m$, procedure Beam-Append with an appropriate parameter setting is a fast and good constructive algorithm.
- For problems with $n/m > 2/3$, the use of an iterative algorithm turns out to be necessary and they improve the results of the constructive algorithms substantially.

Most algorithms presented have already been included into the program package LiSA - A Library of Scheduling algorithms (see http://lisa.math.uni-magdeburg.de), and the remaining ones will be included into the next version.

This research was supported by INTAS (project 03-51-5501).

REFERENCES

Achugbue, J.O. and F.Y. Chin (1982). Scheduling the open shop to minimize mean flow time. *SIAM J. on Computing* **11**, 709–720.

Alcaide, D., J. Sicilia and D. Vigo (1997). A tabu search algorithm for the open shop problem. *Top* **5**, 283–286.

Blum, C. (2005). Beam-aco – hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Comput. Oper. Res.* **32**, 1595–1591.

Bräsel, H. and H. Hennes (2004). On the open-shop problem with preemption and minimizing the average completion time. *European J. Oper. Res.* **157**, 607–619.

Bräsel, H., T. Tautenhahn and F. Werner (1993). Constructive heuristic algorithms for the open-shop problem. *Computing* **51**, 95–110.

Brucker, P., J. Hurink, B. Jurisch and B. Wöstmann (1997). A branch-and-bound algorithm for the open-shop problem. *Discrete Applied Mathematics* **76**, 43–59.

Dorndorf, U., E. Pesch and T. Phan-Huy (2001). Solving the open shop scheduling problem. *Journal of Scheduling* **4**, 157–174.

Gonzalez, S. and T. Sahni (1976). Open shop scheduling to minimize finish time. *J. Assoc. of Comput. Mach.* **23**, 665–679.

Gueret, C., N. Jussien and C. Prins (2000). Using intelligent backtracking to improve branch-and-bound methods: An application to open-shop problems. *European J. Oper. Res.* **127**, 344–354.

Kubiak, W., C. Sriskandarajah and K. Zaras (1991). A note on the complexity of open shop scheduling problems. *INFOR* **29**, 284–294.

Liaw, C.-F. (2000). A hybrid genetic algorithm for the open shop scheduling problem. *European J. Oper. Res.* **124**, 28–42.

Liaw, C.-F., C.-Y. Cheng and M. Chen (2002). The total completion time open shop scheduling problem with a given sequence of jobs on one machine. *Comput. Oper. Res.* **29**, 1251–1266.

Liu, C.Y. and R.L. Bulfin (1987). Scheduling ordered open shops. *Comput. Oper. Res.* **14**, 257–264.

Prins, C. (1994). An overview of scheduling problems arising in satellite communications. *Journal Oper. Res. Soc.* **40**, 611–623.

Queyranne, M. and M. Sviridenko (2002). Approximation algorithms for shop scheduling problems with minsum objective. *Journal of Scheduling* **5**, 287–305.

Werner, F. and A. Winkler (1995). Insertion techniques for the heuristic solution of the job shop problem. *Discrete Appl. Math.* **50**, 191–211.