# Heuristics for Two-Machine Flow Shop Problems with Earliness and Tardiness Penalties

Volker Lauff and Frank Werner
Otto-von-Guericke-Universität, Fakultät für Mathematik,
PSF 4120, 39016 Magdeburg, Germany

January 9, 2003

**Abstract**

We consider a two-machine flow shop problem with a given common due date for all jobs. Penalties are assigned for both early and late completion of jobs, and the objective is the minimization of the total penalty costs. Based on some structural properties, we derive several constructive and iterative algorithms. For asymmetric linear and quadratic penalty functions, these algorithms are compared relative to each other on problems with 40 and 200 jobs as well as with the exact solution for small problems with up to 20 jobs.

**Keywords:** scheduling, flow shop, heuristics, nonregular performance measures

## 1 Introduction

Due to the increasing awareness of the just-in-time production philosophy, there is a need for a generalization of the commonly used objective functions in scheduling. Traditionally, tardiness penalties due to delivery after a contractually arranged due date $d$ are considered. This approach neglects storage costs due to insurance, theft, perishing, and bounded capital for the case of a completion of the job before the due date. Therefore, tardiness together with earliness penalties are considered in order to meet the requests of practice.

Although there are published a lot of results on one-machine and parallel machine problems with earliness and tardiness penalties [1, 2, 6, 7, 13], there are only a few papers dealing with multi-stage processing systems. Two of these papers are devoted to two-machine flow shop problems. In a flow shop, each of $n$ jobs $1, \ldots, n$ has to be processed entirely first on machine $M_1$ and then on $M_2$.

The processing time $p_{ij}$ of each operation $(i, j)$ which represents the processing of job $i$ on machine $M_j$ is given. Once an operation is started, it may not be interrupted. The starting time of the job $i$ on machine $j$ is denoted with $s_{i\,j}$, the completion time by $c_{i\,j} = s_{i\,j} + p_{i\,j}$. The completion time of job $i$ on $M_2$ is denoted with $C_i$. The problem of minimizing the sum of absolute deviations (SAD) from a given common due date $d$ can be written as $F2|d_i = d|\sum |C_i - d|$ in the notation introduced by Graham et al. [4]. Sarper [12] presented a mixed integer formulation for problem $F2|d_i = d|\sum |C_i - d|$, and gave three heuristics which were compared for very small problems with $n \in \{5, 6\}$ jobs with the optimal solution and for small problems with $n \leq 20$ relative to each other.

Gupta et al. [5] developed an enumerative algorithm for problem $F2|d_i = d|\sum f(C_i)$, where $f$ is a general function modelling the earliness and tardiness penalties for the deviation from a given common due date. It is assumed that $f(d) = 0$ and

$$\begin{aligned} &\text{M1: } f(x) \text{ is non-increasing for } x \leq d, \\ &\text{M2: } f(x) \text{ is non-decreasing for } x \geq d. \end{aligned} \qquad (\mathrm{M})$$

Note that this assumption includes regular criteria as a special case, since by setting $f(x) = 0$ for $x \leq d$ arbitrary regular criteria for minimizing total penalty may be modeled. We refer to $f$ as penalty function in order to provide a distinction from the sum of $f$, the objective function. The special case of the SAD-criterion is covered by setting $f(C_i) = |C_i - d|$. Gupta et al. [5] were able to solve instances with up to 20 jobs on a PC for non-symmetric linear and quadratic earliness and tardiness penalty functions. The aim of the present work is to develop and test several heuristic algorithms that are able to generate near-optimal solutions also for larger instances of the problem $F2|d_i = d|\sum f(C_i)$ with $f$ as defined above.

The paper is organized as follows. In Section 2, we cite some results for this problem published in the work on the exact enumeration algorithm [5] and give an extension which is important to develop the heuristics considered in this paper. The constructive algorithms are presented in Section 3, together with a post-optimization procedure. Section 4 is devoted to the description of the iterative algorithms. Section 5 compares the results of the different heuristics obtained for problems with 40 and 200 jobs. Additionally, we compare our solutions with the exact solutions for problems with 14 to 20 jobs obtained by Gupta et al. [5]. Section 6 gives some concluding remarks.

# 2 Basic Concepts and Test Problems

In this section, we give some basic properties and algorithms which are used in the constructive and iterative algorithms for the heuristic solution of the problem. We also describe the generation of the test problems. These are used for the

parameter optimization of the algorithms as well as for the comparative study of the algorithms developed.

## 2.1 Properties of an Optimal Schedule

Gupta et al. [5] gave the following properties of problem $F2|d_i = d|\sum f(C_i)$, which are also valid for the more general case of job-dependent penalty functions $f_i(C_i)$.

**Lemma 1** *The set of all permutation schedules contains an optimal schedule.*

**Proposition 1** *There exists an optimal schedule starting the first job at time 0 and having no idle times on $M_1$.*

**Proposition 2** *There exists an optimal schedule in which the jobs on $M_2$ which are started before or at $d$ have no idle times in-between.*

Additionally, we want to give a direct consequence of Proposition 2. Denote the makespan value (i.e. lowest possible completion time of the last job) of sequence $(\pi_1, \ldots, \pi_i)$ with $C_{\max}(\pi_1, \ldots, \pi_i)$. Define

$$\mu = \min\{i \in \{1, \ldots, n\} | C_{\max}(\pi_1, \ldots, \pi_i) \geq d\}, \tag{1}$$

then the sequence $(\pi_\mu, \ldots, \pi_n)$ constitutes the minimal possible sequence of tardy jobs for the fixed job sequence $\pi = (\pi_1, \ldots, \pi_n)$. Note that, here and in the following, we call a job which completes *exactly* at the due date tardy, too, in order to simplify the presentation. If $C_{\max}(\pi_1, \ldots, \pi_n) < d$ we set $\mu = n$. Obviously, the jobs from $\pi_{\mu+1}$ to $\pi_n$ (always tardy in any feasible schedule for sequence $\pi$) are scheduled as early as the flow shop condition and the jobs scheduled before on $M_2$ allow.

The following proposition strengthens Proposition 2.

**Proposition 3** *Every schedule for the fixed job sequence $\pi = (\pi_1, \ldots, \pi_n)$ with idle times in-between the processing of jobs $\pi_1$ to $\pi_\mu$ can be changed into a schedule without idle time in-between these jobs, such that the objective function value does not increase.*

PROOF: Let $\pi_q$ be the latest job in some schedule for job sequence $\pi$ which is started before the due date $d$. By definition of $\mu$ we have $q \leq \mu$. Due to Proposition 2, the objective function value will not increase, if all idle time is eliminated in-between the jobs $\pi_1, \ldots, \pi_q$ on $M_2$ while keeping $s_{\pi_q 2}$ constant. Likewise, scheduling the tardy jobs of $\pi_{q+1}, \ldots, \pi_\mu$ as early as possible on $M_2$ does not increase the objective function value. By definition of $\mu$ it follows that this can be done without any idle time in-between the jobs $\pi_q, \ldots, \pi_\mu$, since the

starting times of the jobs $\pi_1, \ldots, \pi_\mu$ on $M_1$ do not restrict the scheduling decision. ■

According to Lemma 1 and Propositions 1 and 3, we represent a schedule in the following by the job sequence $\pi$ and the starting time $s_{\pi_1 2}$ of the first job on machine $M_2$.

## 2.2 Optimization for a Given Sequence

The algorithms that we present later perform local search among the set of permutations. This approach requires an algorithm which constructs a *best schedule* for a *given permutation* (i.e. sequence) of the jobs. In this subsection, we present an algorithm for solving this subproblem.

Assume, we are given a permutation $\pi$ of the jobs in set $N$. It remains to determine the best starting time of the first job $\pi_1$ on $M_2$. Denote with

$$\kappa = C_{\max}(\pi_1, \ldots, \pi_\mu) \tag{2}$$

the minimum possible completion time of job $\pi_\mu$, then the interval of potentially optimal starting times for job $\pi_1$ is

$$\left[ \max\{\kappa, d\} - \sum_{i=1}^{\mu} p_{\pi_i 2}, d \right]. \tag{3}$$

In this interval, a modified bisection search is performed as follows.

Denote with $F_E(s_{\pi_1 2})$ and $F_T(s_{\pi_1 2})$ the earliness and tardiness costs of a schedule, respectively, for a given starting time $s_{\pi_1 2}$ and a given permutation $\pi$. We omit $\pi$ as an explicit argument of the cost functions. The following procedure calculates these values $F_E(s_{\pi_1 2})$ and $F_T(s_{\pi_1 2})$.

**Algorithm 1 (Calculate $F_E(s_{\pi_1 2})$, $F_T(s_{\pi_1 2})$)**

Input: permutation $\pi$, starting time $s_{\pi_1 2}$
$F_T(s_{\pi_1 2}) := 0$
$F_E(s_{\pi_1 2}) := 0$
$S2 := s_{\pi_1 2}$
$S1 := 0$
FOR $i := 1$ TO $n$ DO
BEGIN
  $S1 := S1 + p_{\pi_i 1}$
  $S2 := \max\{S1, S2\} + p_{\pi_i 2}$
  IF $S2 < d$ THEN
    $F_E(s_{\pi_1 2}) := F_E(s_{\pi_1 2}) + f(S2)$
  ELSE
    $F_T(s_{\pi_1 2}) := F_T(s_{\pi_1 2}) + f(S2)$

END
RETURN $(F_E(s_{\pi_1 2}), F_T(s_{\pi_1 2}))$

With the arguments $l$ (left), $m$ (middle), and $r$ (right) we distinguish the smallest, medium respectively largest value of $s_{\pi_1 2}$ in some interval. It follows that $F_E(r) \leq F_E(m) \leq F_E(l)$ and $F_T(l) \leq F_T(m) \leq F_T(r)$ (for a fixed sequence $\pi$). Hence, $F_E(r) + F_T(m)$ is a lower bound for the objective function value of the solutions with starting time $s_{\pi_1 2} \in [m, r]$. Analogously, we have the lower bound $F_E(m) + F_T(l)$ in the interval $[l, m]$. Note that for this estimate Condition (M) is sufficient; we do not need a convexity property of the penalty function. Thus we can use a bisection search, where a subinterval can only be excluded from the further search if the corresponding lower bound is greater than or equal to the best function value $F^*$ obtained so far. The bisection procedure calls the procedure above in order to calculate the costs for earliness and tardiness. The following algorithm sketches the procedure.

**Algorithm 2 (Optimization of $s_{\pi_1 2}$)**

Input $(l, r, F_T(l), F_E(r))$
$m = (l + r)/2$
calculate function values $F_E(m), F_T(m)$
IF $(F_E(m) + F_T(m) < F^*)$ THEN
BEGIN
  $F^* = F_E(m) + F_T(m)$
  $s = m$
END
IF $(l >= r - 1)$ RETURN
IF $(F_T(l) + F_E(m) < F^*)$ THEN Optimization for $(l, m, F_T(l), F_E(m))$
IF $((F_T(m) + F_E(r) < F^*)$ THEN Optimization for $(m, r, F_T(m), F_E(r))$

The algorithm is first called with $l = \max\{\kappa, d\} - \sum_{i=1}^{\mu} p_{\pi_i 2}$ and $r = d$ and the (global) variable $F^*$ is initialized with $F^* = \min\{F(l), F(r)\}$.

It is a characteristic of heuristics that most of the computational effort is spent for calculating the objective function value of various solutions. We have found that the optimization procedure above becomes very time-consuming for "loose" (high) due dates and iterative heuristics, whereas the optimal time $s_{\pi_1 2}$ from one iteration to the other rarely changes. Thus, it is reasonable to save computation time by performing the optimization described in Algorithm 2 after some iterations only. In the time in-between two updates of $s_{\pi_1 2}$, we use $\max\{s_{\pi_1 2}, \kappa - \sum_{i=1}^{\mu} p_{\pi_i 2}\}$. For the instances with 200 jobs (see Section 2.3), we updated $s_{\pi_1 2}$ every tenth iteration. In the constructive algorithms, we certainly used Algorithm 2 in every step of the construction.

Compared to the algorithm given in [12], there are the following differences. Sarper does not attempt to remove or insert idle time between jobs as is done in this paper by calling Algorithm 1 for all the different values of $s_{\pi_1 2}$ considered in the course of the bisection search of Algorithm 2. We observed that this is a crucial point in order to find good heuristic solutions. Further, we apply a bisection search instead of examining every reasonable integer starting time $s_{\pi_1 2}$, which significantly reduces the computational effort. Note that in the approach chosen by Sarper only *two* possible starting times are possible anyway for the SAD-criterion which he studied. Since the algorithm in [12] starts with the schedule providing the lowest starting times on $M_2$ possible for a given sequence, there are only two possibilities. Either, the relative value of the due date is low compared with the other input data, then this schedule is already the optimal one for this sequence, or, the due date is rather loose, then the schedule with $s_{\pi_{\lceil n/2 \rceil} 2} = d$ yields an optimal value for the sequence. This follows by a similar argument as given in [9] and by the fact that the algorithm of Sarper does not remove idle times.

## 2.3 Test Problems

The problem data of one instance consists of the number of jobs, the processing times of all operations and the common due date. The job processing times are uniformly distributed integers from the interval $[1, p_{\max}]$. For the problems with 40 and 200 jobs we used $p_{\max} = 100$. For the instances with 14 to 20 jobs we used $p_{\max} \in \{5, 20, 50\}$ as considered in [5]. The common due date was taken to be a function of the maximum load $P$ on the two machines given by:

$$P = \max \left\{ \sum_{i \in N} p_{i\,1}, \sum_{i \in N} p_{i\,2} \right\}. \tag{4}$$

To test the influence of the relative value of the due date compared with the other input data, the following values of the common due date were used:

$$d \in \{0, 0.1P, 0.25P, 0.5P, 0.75P, 1P\}$$

In our tests, we consider the same two objective functions as in [5]:

- an asymmetric linear-linear penalty function:

$$f(x) = \begin{cases} d - x - t, & x < d - t \\ 0, & x \in [d - t, d] \\ 5(x - d), & x > d \end{cases} \tag{5}$$

  with a time window of width $t = 0.25d$;

- a linear-quadratic penalty function:

$$f(x) = \begin{cases} d - x, & x \leq d \\ (x - d)^2, & x > d \end{cases}. \tag{6}$$

# 3 Constructive Algorithms

The objective of constructive algorithms is to generate a good heuristic solution in polynomial time. Several of these approaches are presented in Section 3. The resulting schedules obtained by means of these methods are improved by a smoothing procedure which is described in Section 3.2. We will compare the results obtained by the various methods in Section 3.3.

## 3.1 Construction of a Job Sequence

We studied priority rules as well as insertion techniques. Priority rules are applied to sort the jobs in a list, e.g. according to non-decreasing processing times (SPT-order) on $M_1$. The sequence is built up by appending the next job of the list to the end of the current partial sequence (algorithm Append). Insertion techniques are very common for constructive heuristics and were successfully applied to many scheduling problems (see e.g. [11]). As in the appending methods, the jobs are ordered in a list according to a certain criterion. The sequence is built up successively by inserting the jobs into a partial sequence according to this order. There are $i + 1$ possible positions for a job which is inserted into a sequence consisting of $i$ jobs. Among these $i + 1$ resulting sequences, the one with the lowest objective function value is selected. The underlying idea is that an optimal decision for a part of the sequence is at least a good decision for the whole sequence.

For the problems considered in this paper, we have studied an alternative to the standard insertion algorithms described above. This is motivated by the generality of the penalty function defined in (M). For instance, asymmetry is possible, and the functions considered in this paper are asymmetric (see Equations (5) and (6)): An early deviation from the due date is penalized less than a tardy one with the same absolute value. For an insertion according to the SPT-order, this means that nearly all jobs are sequenced early at the beginning with the consequence that the jobs with larger processing times are sequenced tardy, when there are no more possibilities to schedule them early. This contradicts the idea behind the insertion heuristics, since the optimal decision for a partial sequence is not advantageous from the global point of view. Informally speaking, the standard insertion heuristic recognizes the difficulty too late.

To overcome this problem, we have changed the due date $d$ used in the calculation procedure in each step. Let the current sequence consist of $i$ jobs, and the next job according to the list enters the sequence. Then the due date in this $(i+1)$-th insertion step is $d_{i+1} = (i+1)d/n$, where $n$ is the total number of jobs. This procedure will be referred to as *due date modification*.

## 3.2 Procedure SMOOTH

We will now present a procedure called SMOOTH for post-optimization of a sequence $\pi$ obtained by one of the above constructive algorithms. The motivation stems from a result obtained by Kahlbacher [8] for one-machine problems and the same type of penalty functions fulfilling (M) as considered in this paper. For one-machine problems, a sequence is called V-shaped, if there are no three jobs $i, j, k$, where $i$ is processed before $j$ and $j$ before $k$, with $p_i < p_j$ and $p_k < p_j$. Transferred to the flow shop problem which we consider, this means that the early jobs should strive for an LPT-order (i.e. longest processing time first), and the tardy jobs for an SPT-order with respect to the processing times on $M_2$. Deviations from the strict V-shape sequence on $M_2$ may reduce the objective function value due to the flow shop constraint. These observations lead to the following criterion for interchanging two adjacent jobs $\pi_i$ and $\pi_{i+1}$ in a permutation. An adjacent pairwise interchange of both jobs *does not increase* the objective function value if there are either adjacent

- early jobs $\pi_i, \pi_{i+1}$ with $p_{\pi_i\,2} < p_{\pi_{i+1}\,2}$ or
- tardy jobs $\pi_i, \pi_{i+1}$ with $p_{\pi_i\,2} > p_{\pi_{i+1}\,2}$;

and the inequality

$$c_{\pi_{i+1}\,2} \geq \max(c_{\pi_i\,1} + p_{\pi_{i+1}\,2}, c_{\pi_{i+1}\,1}) + p_{\pi_i\,2}$$

holds. This is due to the fact that the sum of the contributions to the objective functions does not increase by interchanging jobs $\pi_i$ and $\pi_{i+1}$ and the remaining jobs may be processed without altering their starting times. Note that we cannot be sure that the objective function value is indeed *reduced* due to the generality of the penalty function as given in Condition (M).

SMOOTH works as follows. First, the starting time $s_{\pi_1\,2}$ is optimized by Algorithm 2. By this, we can identify the early and tardy jobs of the sequence. Then, we check the conditions given above successively for all indices $i \in \{1, \ldots, n-1\}$ of permutation $\pi$ and perform adjacent pairwise interchanges if one condition is fulfilled. We optimize the scheduling of the resulting modified sequence again by Algorithm 2 and repeat these steps until we get a sequence where both two conditions are not fulfilled for any two adjacent jobs.

## 3.3 Comparison of the Constructive Algorithms

In total, we studied 12 constructive heuristics and a randomly generated sequence sequence for comparison (RAND). We use the following notation in order to distinguish the methods. The first letter is either an I (Insertion) or an A (Append). We append an S, if procedure SMOOTH is applied. The name is accomplished by two acronyms in brackets. The first represents the order according

to which the jobs are considered in the construction process, for instance SPT1 stands for the SPT-order with respect to the processing times on $M_1$. If the due date modification is applied, the second expression in brackets is an D, and N otherwise. Thus, as an example, I(LPT2,N) means that the jobs are inserted according to non-decreasing processing times on $M_2$ without due date modification and without procedure SMOOTH. The abbreviations LL and LQ indicate the linear-linear and the linear-quadratic penalty function, respectively, as given in Equations (5) and (6). For a selection of the tested heuristics, the results for the problems for the two penalty functions with 40 and 200 jobs are given in Tables 1 and 2. For every value $d/P$, we have generated 400 instances. For each heuristic, the first number gives the average percentage deviation from the best function value obtained in the tests of these constructive algorithms (here and in the following tables, all percentage values are rounded down with a precision of two digits behind the decimal point) and the second value (in brackets) gives the number of instances in which the corresponding heuristic has obtained the best value (among the 13 constructive heuristics considered). Since there may be more than one heuristic reaching the best value, the sum over all these values for a fixed $d/P$ can exceed 400 (since only a selection of heuristics is presented, the sum over these values can also be smaller than 400).

INSERT TABLES 1-2 HERE

Comparing the results without and with procedure SMOOTH, we have found that the results with procedure SMOOTH are significantly better. In Tables 1 and 2 we therefore focus on the results for the methods using SMOOTH. We only mention that without SMOOTH, the best methods from an overall point of view are I(SPT1,D), I(SPT2,D), I(SPT1,N), and I(SPT2,N) and that in this case the appending methods are not competitive with the insertion algorithms. To illustrate the differences to the methods using procedure SMOOTH, we give the results for I(SPT2,D) and I(SPT2,N).

With procedure SMOOTH, among the insertion algorithms IS(SPT1,D), IS(SPT2,D), IS(SPT1,N) and IS(SPT2,N) perform best. It can be observed that for $d/P = 0.75$, the methods with due date modification are superior, whereas, for $d/P \leq 0.25$ and 200 jobs, the versions without modification yield better results. For the LQ penalty function, insertion methods basing on SPT1 are superior while for the LL function, insertion methods basing on SPT2 often yield better results. The most astonishing result is the excellent performance of the fast appending procedure AS(SPT1) (which is in strong contrast to the corresponding variant without SMOOTH). The improvements obtained by procedure SMOOTH are so substantial that AS(SPT1) frequently obtains even the best results among all methods, mainly for 200 jobs and low values of $d/P$. We also mention that the most impressive improvements are obtained by procedure SMOOTH for $d/P = 1.0$

Finally we give some comments on the computational times. Without proce-

dure SMOOTH, the required computation time is increasing with the due date because of the broadening of the search interval given in (3) for the optimization of $s_{\pi_1 2}$. Nevertheless, the appending methods show only a slight ascent of computation time needed for increasing values of $d/P$. For $d = 0$ and 200 jobs, the total computation time is 80 seconds, and for $d/P = 1.0$, it is 135 seconds for solving 400 instances, independently of the objective function.

The computation time needed by the insertion methods (without SMOOTH) compared with the appending procedures depends on the ratio $d/P$. We will only discuss the 200 job examples, since the differences are more distinct in this case. From $d = 0$ up to $d/P = 0.25$, the computation time required by the insertion methods increases moderately and is about twice the time for appending methods. For higher ratios $d/P$, we observe different running times for the methods with and without due date modification as well as for the different objective functions considered. We will first discuss the LL penalty function. The insertion algorithms with due date modification need twice as much time for $d/P = 0.50$ as the appending methods, whereas without due date modification, this factor can reach 20. For $d/P = 0.75$, the computation time is factor 20 higher for the insertion methods with due date modification. Without due date modification, this factor can even reach a value of 90. The reason for this different behaviour of the methods with and without due date modification lies in the optimization of the starting time $s_{\pi_1 2}$. If the due date is not modified, the interval given in (3) is larger than with due date modification unless all jobs are inserted in the partial sequence, since the highest value of the interval is $d$ without and $d_{i+1} = (i+1)d/n$ with due date modification if the partial sequence contains $i$ jobs. For $d/P = 1.0$, both methods need factor 100 more than the appending methods. For the same ratio of $d/P$ with the LQ penalty function, we observe the same tendencies, but the factors are much smaller. This shows that the bisection search used to optimize the starting time $s_{\pi_1 2}$ can cut off more intervals than in the case of the LL penalty function.

The computation time required by procedure SMOOTH does not depend significantly on the constructive heuristic. The additional computational effort is negligible in the case of the insertion methods. For the appending methods, the computation time is twice as high with the smoothing procedure.

# 4 Iterative Algorithms

In this section, we deal with the development of iterative algorithms for the approximate solution of the problem under consideration. As mentioned before, the search is performed among the set of permutations. First, we shortly describe the neighbor generation of a permutation. Then we present the metaheuristics that we used in our local search algorithms. Iterative improvement (IT) is used to find a reasonable number of iterations which is applied for all iterative algorithms.

After this, the parameters for simulated annealing (SA) — the cooling scheme and the initial temperature — are optimized.

## 4.1 Neighborhoods

The neighborhoods are based on the known pairwise interchange (PI) and shift (SH) neighborhoods. A pairwise interchange neighbor of a permutation $\pi = (\pi_1, \ldots, \pi_n)$ is generated by interchanging two arbitrary jobs, i.e. a sequence of the form

$$\pi^{PI} = (\pi_1, \ldots, \pi_{i-1}, \pi_j, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_i, \pi_{j+1}, \ldots, \pi_n).$$

Shift neighbors fall into two classes. A left shift neighbor of a permutation is generated by removing an arbitrary job from the sequence and reinserting it at an earlier position, i.e.

$$\pi^{SH} = (\pi_1, \ldots, \pi_{i-1}, \pi_j, \pi_i, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_{j+1}, \ldots, \pi_n).$$

Analogously, a right shift neigbor is obtained by reinserting the removed job at a later position, i.e.

$$\pi^{SH} = (\pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_i, \pi_j, \pi_{j+1}, \ldots, \pi_n).$$

The number of neighbors in the PI neighborhood is $n(n-1)/2$ and the number of neighbors in the SH neighborhood is $(n-1)^2$.

In the local search algorithms that we discuss later, we use the union of both the PI and the SH neighborhood. As we describe in the following subsection, the generation of neighbors is controlled by a parameter $\mathcal{P}_{\mathrm{PI}}$, which gives the probability of generating a PI neighbor. Thus, a (left or right) SH neighbor is generated with the probability $\mathcal{P}_{\mathrm{SH}} = 1 - \mathcal{P}_{\mathrm{PI}}$.

## 4.2 Iterative Improvement

The IT which we use works as follows. We start with an initial permutation. The neighbors of the permutation are scanned in a predefined order. As soon as a permutation with a lower objective function value is found, this permutation is *accepted*, which means that the neighbors of the new permutation are checked now. This goes on until a local minimum is reached, which means that the algorithm has found one permutation which does not have any neighbor with a lower objective function value.

When applying IT, a local minimum should be reached within a small number of iterations, e.g. the convergence should be good. On the other hand, the local minimum obtained should have a low value. Since we accept the first neighbor providing an improvement, the way of scanning the neighborhood may influence the performance of the algorithm substantially.

IT has the probability $\mathcal{P}_{\mathrm{PI}}$ of the execution of the operation PI as the only parameter. $\mathcal{P}_{\mathrm{PI}}$ has been optimized by the following investigation according to the above criteria. As a measure for the convergence we use the variable STEPNUM. This is the average number of iterations required to reach a function value within a deviation of 0.5% of the local optimum finally obtained. Using a random initial sequence, we tested the 11 values for $\mathcal{P}_{\mathrm{PI}}$, from $\mathcal{P}_{\mathrm{PI}} = 0$ to $\mathcal{P}_{\mathrm{PI}} = 1$ in steps of 0.1, and used all four combinations of job number (40, 200) and penalty function (LL, LQ). The values of $d/P$ have been 0.1, 0.5, and 1.0. We have evaluated the results of 400 instances each with respect to the average percentage improvement, the number of best objective function values obtained, and the average and maximal values of STEPNUM.

Summarizing the results of this investigation, we can say that the differences due to the choice of parameter $\mathcal{P}_{\mathrm{PI}}$ are much bigger for the instances with 200 jobs. $\mathcal{P}_{\mathrm{PI}} = 0$ yields significantly bad results for $d/P = 0.1$ and $d/P = 0.5$, whereas other values of $\mathcal{P}_{\mathrm{PI}}$ do not lead to big differences in the objective function value. Especially, the pure PI neighborhood ($\mathcal{P}_{\mathrm{PI}} = 1.0$) does not perform much worse than the union of the SH and the PI neighborhoods. Nevertheless, there are significant differences in the average value of STEPNUM. Using the pure SH neighborhood ($\mathcal{P}_{\mathrm{PI}} = 0$), one finds the local minimum faster than with any other method, but, as mentioned before, the objective function value of this local minimum is much bigger than that of other choices of $\mathcal{P}_{\mathrm{PI}}$. The highest average values for STEPNUM are needed for $\mathcal{P}_{\mathrm{PI}} = 0.1$ and $d/P = 0.1$ and $d/P = 0.5$. The interpretation is obvious: Since the probability of generating a pairwise interchange neighbor is low, many iteration steps are spent with generating shift neighbors which do not lead to an improvement. The average value of STEPNUM for the pure PI neighborhood ($\mathcal{P}_{\mathrm{PI}} = 1.0$) is not much lower than for lower values of $\mathcal{P}_{\mathrm{PI}}$. According to our results, we have decided to use $\mathcal{P}_{\mathrm{PI}} = 0.7$.

We have realized IT in the following way. We first generate two lists which contain all possible operations to generate a pairwise interchange and shift neighbor, respectively. These lists are shuffled randomly, in order to screen the neighborhood with a uniform probability. According to the probability parameter $\mathcal{P}_{\mathrm{PI}}$, the next operation is chosen from the list for the PI or the SH operations. If all sequences of one of the two neighborhoods have been scanned completely, the probability for choosing a neighbor from the other list is 1. If all neighbors have been scanned without finding a better solution, a local minimum has been attained. If a list has been worked through several times, it is shuffled again. Note that this can be done only when a new solution has been accepted, since otherwise some operations could be done twice, once before and once after this shuffling, and other operations could never be done at all. For comparison, we have studied the effects on convergence and on the value of the local minimum if the lists are not shuffled for 200 jobs and the LL penalty function. For $d = 0$, the algorithm without shuffling needed 38 % more iterations on average and the local minimum found was 0.07 % higher. We have observed the highest deviation of the

local minimum for $d/P = 0.50$ with 0.24 %. The highest percentage of additional number of iterations for finding a local minimum was 66 % and has been needed for $d/P = 0.25$. For $d/P = 1.0$, we have not detected a difference between the local minima with and without shuffling. Nevertheless, without shuffling, 25 % additional iterations have been needed. These observations demonstrate the superiority of the suggested approach.

Due to the results, we have chosen IN=100,000 iterations for job numbers from 14 to 40. For 200 jobs, we used IN=2,000,000 iterations. As initial permutations for IT we applied the sequences constructed by the algorithms RAND, I(SPT1,D), I(SPT2,D) and A(SPT1). In order to distinguish these algorithms, we write IT-RAND etc.

## 4.3 Multi-Descent

Multi-descent (MD) is a metaheuristic which applies IT repeatedly with different starting solutions. Often, the number of starting solutions is fixed. In this case, the total number of iterations can vary, since the number of iterations required by IT to obtain a local optimum can vary as well. In our approach, MD is applied until the number of total iterations is reached, which is the only free paremeter of MD. Thus, the number of starting solutions investigated for one instance can vary.

## 4.4 Simulated Annealing

Among the great variety of metaheuristics, simulated annealing (SA) is widely regarded as very suitable and effective (see, for instance, [3] and the literature cited there). In this metaheuristic, a neighbor $S'$ of a schedule $S$ is accepted with a certain probability

$$\rho = \begin{cases} 1, & \delta \leq 0 \\ e^{-\delta/T}, & \delta > 0 \end{cases}, \tag{7}$$

where $\delta = F(S') - F(S)$ is the difference of the two objective function values. Thus, improvements are always accepted whereas neighbors with a higher objective function value are accepted with a probability which decreases sensitively according to the worsening. The parameter $T$ is called temperature, reminding of the original motivation of SA by statistical physics. The higher the value of $T$, the higher is the probability of accepting a certain worsening. The temperature is reduced in the course of the search for a minimum from an initial value TI down to a final value TF. This process is known as cooling.

We apply the Lundy-Mees cooling scheme [10], which means that the reduction of the temperature is carried out in discrete steps according to the following

formula:

$$T^n = \frac{T^o}{1 + \beta T^o}, \tag{8}$$

where $T^o$ is the old and $T^n$ is the new temperature, and

$$\beta = \frac{\mathrm{TI} - \mathrm{TF}}{(\mathrm{IN} - 1)\mathrm{TI} \cdot \mathrm{TF}}. \tag{9}$$

TF has to be a positive number. For a given value IN, only two parameters remain to be free, the initial temperature TI and the final temperature TF.

In order to illustrate the optimization process of these parameters, we give the different combinations considered for problems with the LQ penalty function, 200 jobs, and $d = 0.1$ in Table 3. APD gives the average percentage deviation from the best value in these tests. NBS is the number of instances for which the best solution has been obtained by the particular variant. We used 100 instances for these preliminary tests. First, we have investigated the influence of TF, keeping TI=4,000,000 constant, which has been the best value found in our preceding investigation for $d = 0$. TF=25,000 has yielded the best results among the values tested. In a second step, we have tried for TF=25,000 different values of TI. The actual value of parameter TI is of rather little influence. As can be seen in Table 3, we tested some further combinations of TI and TF as well. Table 4 gives the best parameters found for SA for the problems considered in this paper. For the optimization of TI and TF for the problems with 14 to 20 jobs we have chosen $n = 18$ as a compromise.

We have done the following investigation in order to get an impression of the sensitivity of the parameters TI and TF with the LQ penalty function. For the instances with 200 jobs and $d = 0$, we have used the values TI and TF which had been found to be the best for $d/P = 1.0$. The average deviation of the results with the "wrong" parameters is 0.4 %, which means that in this case, SA performs worse than MD and not much better than IT with a random initial sequence. This can be explained easily: If the temperatures TI and TF are chosen too low, SA does not perform much differently from iterative improvement since nearly all worsenings are rejected. We also used the parameters optimized for $d = 0$ for the problems with $d/P = 1.0$. The average deviation was 2.03 %. This means that with this wrong parameter set SA does not even perform better than the constructive method with a *random* initial sequence and procedure SMOOTH for post-optimization. For $d/P = 0.5$ we also studied the influence of applying the parameters found for the 40 job instances on the 200 job problems and vice versa. In the first case, the average deviation from the values with the correct parameter set was 0.29 %, in the latter, we found an average deviation of 0.25 %.

INSERT TABLES 3-4 HERE

# 5 Results of the Comparative Study

In this section, we present a detailed comparison of the developed constructive and iterative heuristics. First, we compare the chosen variants on problems with 40 and 200 jobs relative to each other. Then we compare the heuristics for problems with $n \leq 20$ jobs with the exact solution of the enumerative algorithm given by Gupta et al. [5].

## 5.1 Comparison of the Heuristics Relative to Each Other

INSERT TABLES 5-6 HERE

In Tables 5 and 6, we give the results for the problems with $n \in \{40, 200\}$ both for the LL and the LQ penalty function. For each of the constructive and iterative heuristics, we give the average percentage deviation from the best value obtained and in brackets, we give the number of times a particular variant obtained the best function value. Among the constructive algorithms, we present the results for the algorithms I(SPT2,D), IS(SPT1,D), IS(SPT2,D) and AS(SPT1). Moreover, we consider iterative improvement (IT) with four different initial solutions characterized by the algorithms given by the acronyms following the hyphen (where RAND stands for a random initial solution). Finally, we present the results of the metaheuristics simulated annealing (SA) and multi-descent (MD).

From Tables 6 and 7, we obtain the following results:

1. For all four problem types (combination of value $n$ and the penalty function), algorithms MD and SA performed best. It can be seen that the results of MD are becoming strictly better with increasing ratio of $d/P$. In contrast, the results of SA are becoming worse with increasing ratio of $d/P$ until $d/P = 0.75$.

2. Problems with $d/P = 1.0$ are easy in the sense that almost all iterative algorithms generate the best function value obtained in the experiments.

3. Algorithm IT works best with the bad initial solutions RAND and A(SPT1).

4. Problems with $d/P = 0.5$ and $d/P = 0.75$ are the hardest ones. In particular for problems with $n = 200$ jobs, $d/P = 0.75$ and the LQ penalty function values, no constructive algorithm obtained average deviations from the best value better than some 7 %. In contrast, for problems with $d/P \leq 0.25$ and $d/P = 1.0$, there is at least one constructive algorithm that produces a rather good solution quality.

The computation time needed by SA and MD is comparable. For the instances with 200 jobs, we observe the following. For $d = 0$, the time is some 40 seconds

per instance. The computation time increases with the ratio $d/P$ due to the time needed by the bisection search. For $d/P = 1.0$, the value is about 300 and 120 seconds for the LL and the LQ penalty function, respectively. Again, these differences stem from the different efficiency of the bisection search for the two functions. The IT algorithms need 10 seconds for $d = 0$, independent from the objective function. For $d/P = 1.0$, the time per instance is 36 and 10 seconds for the LL and the LQ penalty function, respectively. In contrast to the other iterative algorithms, IT needs the maximum computation time for the LQ penalty function and a value of $d/P = 0.25$ (some 18 seconds).

## 5.2 Comparison with the Exact Solution

In this section, we compare the results obtained by the constructive and iterative algorithms presented in this paper with the exact solution determined by the enumerative algorithm by Gupta et al. [5]. They test problems with $n \in \{14, 16, 18, 20\}$ and $d/P \in \{0, 0.1, 0.25, 0.5, 0.75, 1\}$. The processing times are integers generated randomly from the interval $[1, p_{\max}]$ with $p_{\max} \in \{5, 20, 50\}$. For any combination of $n, d/P, p_{\max}$, and the two penalty functions (see Equations (5) and (6)), 20 instances are generated, yielding in total 2,880 test problems. In Table 7, the results for two representative combinations of $n$ and the penalty function are given, namely for $n = 14$ and the LL function with $p_{\max} = 5$ and for $n = 20$ and the LQ function with $p_{\max} = 50$. In both cases, the results for 20 instances are given. For each algorithm, we give in the first row the average percentage deviation from the optimal value and the number of times how often the optimal value has been found. Note that the optimality of the best solution obtained in [5] is not proven for all instances. Despite of this fact we do not distinguish between optimal and best solutions obtained by the branch and bound algorithm within the time limit and use the term "optimal" for simplicity. We include the same algorithms as in Tables 5 and 6 and add the results by the branch and bound algorithm (BB) in Gupta et al. [5]. The number of optimal solutions found by this algorithm can be less than 20 in the cases where the algorithm did not prove optimality of the best solution within the time limit.

From Table 7, we can draw the following conclusions:

1. Algorithms SA and MD solved all instances with 14 jobs to optimality. The variants of IT produced solutions with an average percentage deviation from the optimal solution smaller than 2 %. Again, from the results for IT, we see that the problems with $d/P = 0.75$ are the hardest ones.

2. For the problems with 20 jobs, MD and SA give the best results, too, where algorithm MD is slightly superior. Except for problems with $d/P = 0.75$, both algorithms produced results that deviate from the optimal value by no more than 1.2 % on average. The results for the IT variants are inferior.

However, even for $d/P = 0.75$, all IT variants deviated by no more than 3.25 % from the optimal value on average.

3. In general, the quality of the solutions found by the heuristics becomes worse with increasing ratio $d/P$ until $d/P = 0.75$. In contrast, for $d/P = 1.0$, all iterative algorithms generated solutions with a function value very close to the optimum.

INSERT TABLE 7 HERE

# 6 Concluding Remarks

In this paper, we have presented a comprehensive study of heuristics to solve approximately a rather general two-machine flow shop problem with a common due date and the minimization of the total earliness and tardiness penalties as objective. In the field of constructive algorithms, we have introduced a new method for post-optimization which we call smoothing which improves the results significantly. For the constructive algorithms, we have found that the individual algorithms yield average percentage deviations from the best value in a rather large range with at least partially very surprising results, e.g. the rather good performance of the appending algorithm with use of the SMOOTH procedure for small values of $d/P$.

The iterative algorithms were compared to the exact solution for problems with up to 20 jobs and proven to generate near-optimal solutions. For problems with $n = 40$ and $n = 200$ jobs, we have compared the algorithms relative to each other. We have found that for the problems considered, finding the parameter settings for simulated annealing turned out to be very time consuming and that a good parameter setting strongly depends on the particular problem type. In the literature, parameters for SA are usually found for specialized assumptions (e.g. randomly determined processing times from a specific interval). However, as the results of our tests show, the parameters found cannot necessarily be used for other problem data (without further tests). Compared with SA, the MD algorithm generated solutions of approximately the same solution quality. However, the MD algorithm has the advantage that no parameter optimization is required and thus is easily applicable also to other problems of this type. Another topic for future research is to design and evalute tabu search algorithms.

# References

[1] U. Bagchi, R. Sullivan, and Y. Chang. Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties

and a common due date. *Naval Res. Logist. Quart.*, 34:739–751, 1987.

[2] K. R. Baker and G. D. Scudder. Sequencing with earliness and tardiness penalties. *Oper. Res.*, 38:22–36, 1989.

[3] R. W. Eglese. Simulated annealing: A tool for operational research. *Eur. J. Oper. Res.*, 46:271–281, 1990.

[4] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling. *Annals of Disc. Math.*, 5:287–326, 1979.

[5] J. N. D. Gupta, V. Lauff, and F. Werner. A branch and bound algorithm for two-machine flow shop problems with earliness and tardiness penalties. *Otto-von-Guericke-Universität, FMA, Preprint 33/99*, 1999.

[6] N. Hall, W. Kubiak, and S. Sethi. Earliness-tardiness scheduling problems, ii: Deviation of completion times about a restrictive common due date. *Oper. Res.*, 5:847–856, 1991.

[7] N. Hall and M. Posner. Earliness-tardiness scheduling problems, i: Weighted deviation of completion times about a common due date. *Oper. Res.*, 5:836–846, 1991.

[8] H. G. Kahlbacher. Scheduling with monotonous earliness and tardiness penalties. *Eur. J. Oper. Res.*, 64:258–277, 1993.

[9] J. Kanet. Minimizing the average deviation of job completion times about a common due date. *Naval Res. Logist. Quart.*, 28:643–651, 1981.

[10] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Math. Prog.*, 34:111–124, 1986.

[11] M. Nawaz, E. E. Enscore, and I. Ham. A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. *OMEGA*, 11:91–95, 1983.

[12] H. Sarper. Minimizing the sum of absolute deviations about a common due date for the two-machine flow shop problem. *Appl. Math. Modelling*, 19:153–161, 1995.

[13] W. Szwarc. Single machine scheduling to minimize absolute deviation of completion times from a common due date. *Naval Res. Logist. Quart.*, 36:663–673, 1989.

| $d/P$ | 0 | 0.1 | 0.25 | 0.5 | 0.75 | 1.0 |
|---|---|---|---|---|---|---|
| 40 JOBS, $p_{max} = 100$ | | | | | | |
| I(SPT2,D) | 0.75 (7) | 1.14 (0) | 1.88 (2) | 3.03 (12) | 1.92 (41) | 0.35 (128) |
| I(SPT2,N) | 0.75 (7) | 1.96 (1) | 2.77 (1) | 5.13 (9) | 5.39 (43) | 0.32 (57) |
| IS(SPT1,D) | 0.54 (119) | 0.95 (51) | 1.67 (33) | 3.12 (23) | 4.13 (35) | 0.33 (120) |
| IS(SPT2,D) | 0.48 (140) | 0.73 (100) | 0.90 (126) | 1.28 (163) | 1.23 (169) | 0.32 (132) |
| IS(LPT1,D) | 3.01 (2) | 4.37 (1) | 7.22 (0) | 11.17 (0) | 7.89 (0) | 1.64 (25) |
| IS(LPT2,D) | 6.29 (0) | 7.93 (0) | 10.38 (0) | 12.77 (7) | 10.02 (11) | 6.34 (2) |
| IS(SPT1,N) | 0.54 (119) | 0.72 (51) | 1.45 (33) | 3.62 (23) | 4.55 (35) | 0.76 (120) |
| IS(SPT2,N) | 0.48 (140) | 1.54 (48) | 2.15 (71) | 4.67 (83) | 5.30 (65) | 0.32 (57) |
| IS(LPT1,N) | 3.01 (2) | 6.97 (0) | 16.86 (0) | 22.66 (1) | 12.71 (7) | 1.00 (36) |
| IS(LPT2,N) | 6.29 (0) | 8.52 (0) | 13.99 (0) | 19.53 (0) | 14.54 (11) | 7.13 (0) |
| AS(SPT1) | 0.78 (137) | 1.25 (91) | 2.23 (80) | 6.29 (30) | 9.66 (6) | 7.42 (3) |
| AS(SPT2) | 34.24 (0) | 42.06 (0) | 53.00 (0) | 70.29 (0) | 58.58 (0) | 18.16 (0) |
| 200 JOBS, $p_{max} = 100$ | | | | | | |
| I(SPT2,D) | 0.43 (0) | 1.32 (0) | 3.64 (0) | 6.28 (0) | 3.52 (0) | 0.01 (197) |
| I(SPT2,N) | 0.43 (0) | 0.87 (0) | 1.20 (0) | 2.45 (1) | 5.98 (9) | 0.00 (136) |
| IS(SPT1,D) | 0.27 (126) | 1.07 (1) | 2.71 (0) | 3.91 (0) | 2.36 (48) | 0.01 (59) |
| IS(SPT2,D) | 0.32 (63) | 1.00 (3) | 2.42 (0) | 3.07 (11) | 1.44 (86) | 0.01 (206) |
| IS(LPT1,D) | 4.28 (0) | 7.43 (0) | 12.75 (0) | 13.51 (0) | 5.05 (2) | 0.59 (0) |
| IS(LPT2,D) | 9.24 (0) | 12.12 (0) | 17.37 (0) | 20.28 (0) | 11.73 (1) | 3.25 (0) |
| IS(SPT1,N) | 0.27 (126) | 0.78 (7) | 1.61 (12) | 1.85 (48) | 1.91 (118) | 0.18 (1) |
| IS(SPT2,N) | 0.32 (63) | 0.65 (14) | 0.82 (100) | 2.21 (153) | 5.85 (41) | 0.00 (136) |
| IS(LPT1,N) | 4.28 (0) | 11.33 (0) | 16.57 (0) | 25.81 (0) | 16.73 (0) | 0.38 (0) |
| IS(LPT2,N) | 9.24 (0) | 13.73 (0) | 21.79 (0) | 25.60 (0) | 13.80 (0) | 4.12 (0) |
| AS(SPT1) | 0.23 (211) | 0.12 (60) | 0.22 (287) | 1.46 (188) | 2.30 (104) | 1.65 (0) |
| AS(SPT2) | 35.72 (0) | 43.39 (0) | 56.67 (0) | 76.16 (0) | 50.71 (0) | 6.31 (0) |

Table 1: Comparison of the constructive methods for the LL penalty function

| $d/P$ | 0 | 0.1 | 0.25 | 0.5 | 0.75 | 1.0 |
|---|---|---|---|---|---|---|
| 40 JOBS, $p_{max} = 100$ | | | | | | |
| I(SPT2,D) | 2.87 (0) | 3.65 (0) | 4.53 (0) | 5.42 (0) | 8.08 (0) | 5.02 (7) |
| I(SPT2,N) | 2.87 (0) | 3.52 (1) | 4.73 (0) | 10.20 (1) | 23.55 (1) | 3.42 (8) |
| IS(SPT1,D) | 1.93 (54) | 2.54 (17) | 3.05 (34) | 2.64 (122) | 1.46 (194) | 3.41 (42) |
| IS(SPT2,D) | 2.52 (17) | 3.26 (9) | 4.07 (15) | 4.78 (23) | 7.38 (15) | 2.30 (32) |
| IS(LPT1,D) | 8.48 (0) | 10.61 (0) | 14.82 (0) | 20.90 (0) | 18.49 (40) | 1.53 (32) |
| IS(LPT2,D) | 11.29 (0) | 12.97 (0) | 14.94 (1) | 15.59 (13) | 13.44 (33) | 5.86 (2) |
| IS(SPT1,N) | 1.93 (54) | 2.11 (31) | 2.50 (39) | 3.90 (82) | 11.23 (62) | 4.89 (55) |
| IS(SPT2,N) | 2.52 (17) | 2.94 (18) | 3.88 (26) | 9.04 (17) | 22.23 (2) | 3.16 (10) |
| IS(LPT1,N) | 8.48 (0) | 13.24 (1) | 23.71 (0) | 39.28 (0) | 46.91 (0) | 0.44 (118) |
| IS(LPT2,N) | 11.29 (0) | 13.17 (0) | 18.66 (0) | 27.43 (0) | 40.99 (1) | 8.17 (176) |
| AS(SPT1) | 0.19 (322) | 0.23 (316) | 0.46 (272) | 2.57 (137) | 9.46 (31) | 10.80 (0) |
| AS(SPT2) | 67.04 (0) | 78.65 (0) | 100.60 (0) | 159.80 (0) | 262.30 (0) | 122.00 (0) |
| 200 JOBS, $p_{max} = 100$ | | | | | | |
| I(SPT2,D) | 3.40 (0) | 4.53 (0) | 6.22 (0) | 7.41 (0) | 5.88 (0) | 1.17 (0) |
| I(SPT2,N) | 3.40 (0) | 3.40 (0) | 3.34 (0) | 3.82 (0) | 9.58 (0) | 0.61 (0) |
| IS(SPT1,D) | 3.17 (1) | 4.29 (0) | 5.87 (0) | 6.68 (1) | 3.78 (71) | 0.91 (15) |
| IS(SPT2,D) | 3.15 (0) | 4.26 (0) | 5.92 (0) | 7.00 (0) | 5.34 (14) | 0.51 (0) |
| IS(LPT1,D) | 11.68 (0) | 15.52 (0) | 23.29 (0) | 35.08 (0) | 36.98 (0) | 0.46 (8) |
| IS(LPT2,D) | 17.94 (0) | 21.21 (0) | 25.62 (0) | 29.42 (0) | 25.56 (4) | 1.41 (0) |
| IS(SPT1,N) | 3.17 (1) | 3.29 (1) | 3.27 (2) | 2.99 (5) | 5.07 (86) | 0.10 (103) |
| IS(SPT2,N) | 3.15 (0) | 3.05 (0) | 2.87 (8) | 3.22 (70) | 8.89 (49) | 0.54 (0) |
| IS(LPT1,N) | 11.68 (0) | 21.03 (0) | 35.24 (0) | 60.23 (0) | 82.03 (0) | 0.01 (180) |
| IS(LPT2,N) | 17.94 (0) | 22.54 (0) | 30.64 (0) | 44.95 (0) | 77.27 (0) | 0.03 (99) |
| AS(SPT1) | 0.00 (399) | 0.00 (398) | 0.01 (388) | 0.56 (324) | 2.86 (173) | 1.31 (0) |
| AS(SPT2) | 70.93 (0) | 83.86 (0) | 109.60 (0) | 187.70 (0) | 304.50 (0) | 10.48 (0) |

Table 2: Comparison of the constructive methods for the LQ function

| TI | TF | APD | NBS |
|---|---|---|---|
| 4,000,000 | 25,000 | 0.02 | 16 |
| 4,000,000 | 5,000 | 0.11 | 0 |
| 4,000,000 | 50,000 | 0.05 | 1 |
| 4,000,000 | 15,000 | 0.03 | 12 |
| 4,000,000 | 2,500 | 0.19 | 0 |
| 1,000,000 | 50,000 | 0.06 | 0 |
| 1,000,000 | 15,000 | 0.04 | 8 |
| 2,000,000 | 15,000 | 0.03 | 8 |
| 3,000,000 | 15,000 | 0.03 | 10 |
| 1,000,000 | 250,000 | 0.52 | 0 |
| 5,000,000 | 250,000 | 0.52 | 0 |
| 2,000,000 | 25,000 | 0.03 | 8 |
| 2,500,000 | 25,000 | 0.02 | 9 |
| 3,000,000 | 25,000 | 0.03 | 9 |
| 5,000,000 | 25,000 | 0.03 | 13 |
| 7,500,000 | 25,000 | 0.02 | 9 |

Table 3: Determination of appropriate values of TI and TF for SA for the LQ penalty function with $d = 0.1$ and 200 jobs

| d/P | 0 | | 0.1 | | 0.25 | | 0.5 | | 0.75 | | 1.0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PARAMETERS FOR 18 JOBS, $p_{\max} = 5$, LL | | | | | | | | | | | | |
| TI, TF | 4 | 4 | 4 | 4 | 4 | 4 | 2 | 2 | 1 | 1 | 1 | 1 |
| PARAMETERS FOR 18 JOBS, $p_{\max} = 20$, LL | | | | | | | | | | | | |
| TI, TF | 10 | 10 | 10 | 10 | 10 | 10 | 5 | 5 | 2 | 2 | 2 | 2 |
| PARAMETERS FOR 18 JOBS, $p_{\max} = 50$, LL | | | | | | | | | | | | |
| TI, TF | 20 | 20 | 20 | 20 | 20 | 20 | 10 | 10 | 3 | 3 | 3 | 3 |
| PARAMETERS FOR 40 JOBS, $p_{\max} = 100$, LL | | | | | | | | | | | | |
| TI, TF | 5,000 | 15 | 15,000 | 15 | 5,000 | 10 | 2,500 | 5 | 1,500 | 1.5 | 1,500 | 1.5 |
| PARAMETERS FOR 200 JOBS, $p_{\max} = 100$, LL | | | | | | | | | | | | |
| TI, TF | 50,000 | 25 | 5,000 | 25 | 500 | 15 | 5,000 | 10 | 500 | 20 | 5,000 | 15 |
| PARAMETERS FOR 18 JOBS, $p_{\max} = 5$, LQ | | | | | | | | | | | | |
| TI, TF | 100 | 20 | 100 | 10 | 10 | 20 | 25,000 | 1 | 300 | 2 | 20 | 0.1 |
| PARAMETERS FOR 18 JOBS, $p_{\max} = 20$, LQ | | | | | | | | | | | | |
| TI, TF | 400 | 400 | 250 | 250 | 200 | 250 | 2,000 | 5 | 100 | 1.5 | 100 | 0.5 |
| PARAMETERS FOR 18 JOBS, $p_{\max} = 50$, LQ | | | | | | | | | | | | |
| TI, TF | 2,000 | 2,000 | 1,000 | 1,000 | 500 | 1200 | 10,000 | 50 | 500 | 10 | 50 | 1 |
| PARAMETERS FOR 40 JOBS, $p_{\max} = 100$, LQ | | | | | | | | | | | | |
| TI, TF | 30,000,000 | 500 | 100,000,000 | 500 | 100,000,000 | 500 | 10,000,000 | 200 | 100,000 | 10 | 10,000 | 0.1 |
| PARAMETERS FOR 200 JOBS, $p_{\max} = 100$, LQ | | | | | | | | | | | | |
| TI, TF | 4,000,000 | 25,000 | 4,000,000 | 25,000 | 1,500,000 | 25,000 | 300,000 | 10,000 | 150,000 | 500 | 50,000 | 250 |

Table 4: The parameters for SA and different problem types

| $d/P$ | 0 | 0.1 | 0.25 | 0.5 | 0.75 | 1.0 |
|---|---|---|---|---|---|---|
| 40 JOBS, $p_{max} = 100$ | | | | | | |
| | CONSTRUCTIVE ALGORITHMS | | | | | |
| I(SPT2,D) | 3.22 (0) | 4.15 (0) | 5.63 (0) | 7.47 (0) | 5.65 (0) | 0.49 (5) |
| IS(SPT1,D) | 3.02 (0) | 3.95 (0) | 5.40 (0) | 7.56 (0) | 7.96 (0) | 0.47 (11) |
| IS(SPT2,D) | 2.96 (0) | 3.73 (0) | 4.61 (0) | 5.64 (0) | 4.93 (0) | 0.46 (5) |
| AS(SPT1) | 3.26 (0) | 4.26 (0) | 5.98 (0) | 10.85 (0) | 13.65 (0) | 7.57 (0) |
| | ITERATIVE ALGORITHMS | | | | | |
| IT-RAND | 0.45 (5) | 0.51 (10) | 0.53 (14) | 0.49 (45) | 0.24 (114) | 0.01 (234) |
| IT-I(SPT1,D) | 0.65 (2) | 0.83 (3) | 1.10 (5) | 0.96 (34) | 0.28 (89) | 0.02 (186) |
| IT-I(SPT2,D) | 0.55 (5) | 0.70 (3) | 0.96 (4) | 0.94 (26) | 0.31 (111) | 0.02 (204) |
| IT-A(SPT1) | 0.53 (1) | 0.54 (6) | 0.51 (10) | 0.49 (41) | 0.21 (121) | 0.02 (225) |
| SA | 0.01 (321) | 0.01 (286) | 0.02 (240) | 0.05 (165) | 0.02 (308) | 0.00 (343) |
| MD | 0.12 (73) | 0.12 (102) | 0.08 (145) | 0.02 (297) | 0.00 (374) | 0.00 (400) |
| 200 JOBS, $p_{max} = 100$ | | | | | | |
| | CONSTRUCTIVE ALGORITHMS | | | | | |
| I(SPT2,D) | 4.04 (0) | 5.17 (0) | 7.63 (0) | 12.07 (0) | 8.83 (0) | 0.02 (1) |
| IS(SPT1,D) | 3.87 (0) | 4.90 (0) | 6.67 (0) | 9.57 (0) | 7.63 (0) | 0.01 (0) |
| IS(SPT2,D) | 3.93 (0) | 4.83 (0) | 6.37 (0) | 8.68 (0) | 6.66 (0) | 0.01 (1) |
| AS(SPT1) | 3.84 (0) | 3.91 (0) | 4.08 (0) | 6.97 (0) | 7.52 (0) | 1.66 (0) |
| | ITERATIVE ALGORITHMS | | | | | |
| IT-RAND | 0.48 (0) | 0.42 (0) | 0.42 (0) | 0.51 (2) | 0.11 (41) | 0.00 (141) |
| IT-I(SPT1,D) | 1.09 (0) | 1.33 (0) | 1.69 (0) | 1.56 (1) | 0.14 (15) | 0.00 (141) |
| IT-I(SPT2,D) | 0.97 (0) | 1.22 (0) | 1.60 (0) | 1.51 (0) | 0.14 (25) | 0.00 (160) |
| IT-A(SPT1) | 0.55 (0) | 0.47 (0) | 0.40 (0) | 0.51 (0) | 0.10 (33) | 0.00 (136) |
| SA | 0.00 (400) | 0.00 (400) | 0.00 (400) | 0.00 (390) | 0.13 (4) | 0.14 (0) |
| MD | 0.35 (0) | 0.30 (0) | 0.26 (0) | 0.24 (7) | 0.00 (299) | 0.00 (358) |

Table 5: Comparison of the various heuristics for the LL penalty function

| $d/P$ | 0 | 0.1 | 0.25 | 0.5 | 0.75 | 1.0 |
|---|---|---|---|---|---|---|
| 40 JOBS, $p_{max} = 100$ | | | | | | |
| | CONSTRUCTIVE ALGORITHMS | | | | | |
| I(SPT2,D) | 5.36 (0) | 6.03 (0) | 6.97 (0) | 8.69 (0) | 11.60 (0) | 5.25 (1) |
| IS(SPT1,D) | 4.40 (0) | 4.88 (0) | 5.45 (0) | 5.81 (0) | 4.76 (0) | 3.63 (10) |
| IS(SPT2,D) | 5.00 (0) | 5.62 (0) | 6.49 (0) | 8.02 (0) | 10.86 (0) | 2.53 (3) |
| AS(SPT1) | 2.62 (0) | 2.52 (0) | 2.79 (0) | 5.71 (0) | 12.93 (0) | 11.03 (0) |
| | ITERATIVE ALGORITHMS | | | | | |
| IT-RAND | 0.43 (21) | 0.45 (27) | 0.43 (30) | 0.41 (54) | 0.38 (128) | 0.00 (366) |
| IT-I(SPT1,D) | 0.80 (9) | 0.86 (4) | 0.91 (9) | 0.71 (50) | 0.34 (153) | 0.00 (358) |
| IT-I(SPT2,D) | 0.77 (8) | 0.80 (2) | 0.79 (6) | 0.70 (44) | 0.45 (136) | 0.00 (359) |
| IT-A(SPT1) | 0.51 (17) | 0.50 (18) | 0.48 (21) | 0.45 (46) | 0.30 (149) | 0.00 (374) |
| SA | 0.11 (156) | 0.10 (175) | 0.11 (162) | 0.12 (67) | 0.19 (18) | 0.00 (392) |
| MD | 0.05 (214) | 0.05 (199) | 0.04 (218) | 0.01 (299) | 0.00 (362) | 0.00 (400) |
| 200 JOBS, $p_{max} = 100$ | | | | | | |
| | CONSTRUCTIVE ALGORITHMS | | | | | |
| I(SPT2,D) | 6.48 (0) | 7.29 (0) | 8.58 (0) | 10.49 (0) | 10.89 (0) | 1.20 (0) |
| IS(SPT1,D) | 6.25 (0) | 7.04 (0) | 8.22 (0) | 9.74 (0) | 8.68 (0) | 0.94 (0) |
| IS(SPT2,D) | 6.22 (0) | 7.01 (0) | 8.27 (0) | 10.06 (0) | 10.32 (0) | 0.54 (0) |
| AS(SPT1) | 2.98 (0) | 2.64 (0) | 2.23 (0) | 3.43 (0) | 7.63 (0) | 1.34 (0) |
| | ITERATIVE ALGORITHMS | | | | | |
| IT-RAND | 0.47 (0) | 0.44 (0) | 0.43 (0) | 0.43 (3) | 0.25 (18) | 0.00 (133) |
| IT-I(SPT1,D) | 1.50 (0) | 1.60 (0) | 1.63 (0) | 1.28 (1) | 0.48 (24) | 0.00 (119) |
| IT-I(SPT2,D) | 1.39 (0) | 1.49 (0) | 1.53 (0) | 1.16 (0) | 0.45 (23) | 0.00 (121) |
| IT-A(SPT1) | 0.50 (0) | 0.47 (0) | 0.44 (0) | 0.44 (3) | 0.23 (30) | 0.00 (133) |
| SA | 0.00 (400) | 0.00 (400) | 0.00 (398) | 0.00 (369) | 0.04 (134) | 0.00 (390) |
| MD | 0.34 (0) | 0.29 (0) | 0.26 (2) | 0.20 (28) | 0.03 (207) | 0.00 (390) |

Table 6: Comparison of the various heuristics for the LQ penalty function

| $d/P$ | 0 | 0.1 | 0.25 | 0.5 | 0.75 | 1.0 |
|---|---|---|---|---|---|---|
| 14 JOBS, $p_{max} = 5$, LL | | | | | | |
| | CONSTRUCTIVE ALGORITHMS | | | | | |
| I(SPT2,D) | 1.41 (1) | 1.70 (0) | 2.40 (2) | 5.80 (0) | 5.61 (1) | 4.69 (5) |
| IS(SPT1,D) | 1.96 (2) | 2.38 (1) | 3.42 (0) | 6.26 (0) | 10.17 (1) | 9.29 (1) |
| IS(SPT2,D) | 1.04 (6) | 1.35 (2) | 2.12 (3) | 4.02 (0) | 4.03 (2) | 2.40 (7) |
| AS(SPT1) | 1.09 (4) | 1.89 (2) | 3.80 (0) | 6.28 (0) | 12.54 (0) | 18.06 (0) |
| | ITERATIVE ALGORITHMS | | | | | |
| IT-RAND | 0.31 (12) | 0.24 (13) | 0.80 (12) | 1.15 (11) | 1.82 (11) | 0.54 (17) |
| IT-I(SPT1,D) | 0.33 (12) | 0.43 (11) | 0.58 (13) | 1.30 (9) | 1.28 (9) | 0.31 (17) |
| IT-I(SPT2,D) | 0.23 (14) | 0.26 (14) | 0.32 (14) | 0.60 (12) | 0.67 (12) | 0.34 (17) |
| IT-A(SPT1) | 0.40 (14) | 0.46 (13) | 0.60 (10) | 0.66 (12) | 0.91 (10) | 0.09 (19) |
| SA | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (20) |
| MD | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (20) |
| | ENUMERATIVE ALGORITHM | | | | | |
| BB | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (20) |
| 20 JOBS, $p_{max} = 50$, LQ | | | | | | |
| I(SPT2,D) | 4.75 (0) | 6.11 (0) | 7.50 (0) | 10.62 (0) | 16.44 (0) | 3.95 (2) |
| IS(SPT1,D) | 3.42 (0) | 4.82 (0) | 5.55 (0) | 6.39 (0) | 9.86 (0) | 6.98 (0) |
| IS(SPT2,D) | 4.44 (0) | 5.67 (0) | 7.13 (0) | 10.10 (0) | 16.17 (0) | 3.14 (0) |
| AS(SPT1) | 2.86 (0) | 4.38 (0) | 5.92 (0) | 10.20 (0) | 25.63 (0) | 20.70 (0) |
| | ITERATIVE ALGORITHMS | | | | | |
| IT-RAND | 0.52 (1) | 1.49 (2) | 2.05 (0) | 1.71 (3) | 2.73 (2) | 0.14 (10) |
| IT-I(SPT1,D) | 1.01 (1) | 1.90 (3) | 1.95 (0) | 2.08 (3) | 3.13 (5) | 0.10 (11) |
| IT-I(SPT2,D) | 0.74 (0) | 1.79 (2) | 1.93 (2) | 2.30 (3) | 2.95 (5) | 0.10 (11) |
| IT-A(SPT1) | 0.51 (1) | 1.48 (3) | 1.84 (0) | 1.72 (2) | 3.25 (3) | 0.10 (11) |
| SA | 0.03 (6) | 0.99 (4) | 1.15 (0) | 1.20 (2) | 2.44 (1) | 0.10 (11) |
| MD | 0.01 (7) | 0.96 (5) | 1.13 (4) | 1.19 (5) | 2.36 (8) | 0.10 (11) |
| | ENUMERATIVE ALGORITHM | | | | | |
| BB | 0.00 (13) | 0.00 (15) | 0.00 (20) | 0.00 (20) | 0.00 (20) | 0.00 (19) |

Table 7: Comparison of the various heuristics and the exact algorithm for both objective functions