Metaheuristic Approaches for the Two-Machine Flow-Shop Problem with Weighted Late Work Criterion and Common Due Date

Jacek Błażewicz
Institute of Computing Science, Poznań University of Technology
Piotrowo 3A, 60-965 Poznań, Poland
Jacek.Blazewicz@cs.put.poznan.pl

Erwin Pesch
Institute of Information Systems, FB 5 - Faculty of Economics, University of Siegen
Hölderlinstrasse 3, 57068 Siegen, Germany
pesch@fb5.uni-siegen.de

Małgorzata Sterna
Institute of Computing Science, Poznań University of Technology
Piotrowo 3A, 60-965 Poznań, Poland
Malgorzata.Sterna@cs.put.poznan.pl

Frank Werner
Faculty of Mathematics, Otto-von-Guericke-University
PSF 4120, 39016, Magdeburg, Germany
Frank.Werner@mathematik.uni-magdeburg.de

Abstract: In this paper, metaheuristic approaches for the two-machine flow shop problem with a common due date and the weighted late work performance measure ($F2|d_j=d|Y_w$) are presented. The late work criterion estimates the quality of a solution with regard to the duration of the late parts of jobs, not taking into account the quantity of the delay for the fully late activities. Since, the problem mentioned is known to be NP-hard, three trajectory methods, namely simulated annealing, tabu search and variable neighborhood search are proposed based on the special features of the case under consideration. Then, the results of computational experiments are reported, in which the metaheuristics were compared one to each other, as well to an exact approach and a list scheduling algorithm.

Keywords: scheduling, flow-shop, late work criteria, metaheuristic, tabu search, simulated annealing, variable neighborhood search

1. Introduction

The rapid development of real time systems makes the due date involving criteria especially useful and important, increasing the interest devoted by researchers to this branch of the scheduling theory. The quality of solutions in real systems is usually estimated from different points of view, which can be modeled by different performance measures (cf. e.g. [4, 12, 13, 28]).

The late work criteria are relatively new objective functions, which have not been so intensively explored as the maximum lateness or tardiness ones, for example. They estimate the quality of a solution with regard to the number of tardy units of particular activities executed in the system. The late work concept was introduced in the context of the scheduling problem on identical parallel machines [3] and, then, applied to uniform [5] and single [18, 21, 22, 25, 26, 29, 30] machine cases. Recently, practical motivations have directed the research to the shop environment, i.e. to systems with dedicated machines [6-11, 27, 31]. Moreover, it is worth to be mentioned that the late work idea is strictly related to the imprecise computation model (cf. e.g. [27]), in which tasks are divided into two parts: a mandatory and an optional one.

In this paper, we continue the research [9-11] on the two machine flow shop problem with the weighted late work criterion and the common due date, which is known to be NP-hard, presenting metaheuristics approaches for its solution. In Section 2, the formal definition of the case under consideration is given. In Section 3, the tabu search, simulated annealing and the variable neighborhood search methods are described, while Section 4 contains the results of computational experiments performed for these search strategies. Some conclusions are given in Section 5.

2. Problem Formulation

The two machine flow shop problem with the weighted late work criterion and a common due date, $F2|d_j=d|Y_w$, concerns the scheduling of a set of jobs $J=\{J_1, \ldots, J_j, \ldots, J_n\}$ on two dedicated machines M_1, M_2. Each job J_j has to be performed first on machine M_1 and then on M_2 for p_{1j} and p_{2j} time units, respectively. Each machine can process only one job at any time and, analogously, each job can be executed by only one machine at any time. We look for a non-preemptive schedule minimizing the late work in the system, i.e. minimizing the amount of work executed after a given common due date d.

Denoting by C_{ij} the completion time of the job J_j on machine M_i, the late work Y_j for this job is determined as (cf. Fig. 2.1):

$$Y_j = \sum_{i=1,2} \min\{\max\{0, C_{ij} - d\}, p_{ij}\}$$

The criterion value to be minimized, estimating the quality of a complete schedule for the whole set of n jobs, taking into account their given weights w_j, j=1,...,n, is determined as:

$$Y_w = \sum_{j=1}^{n} w_j Y_j$$

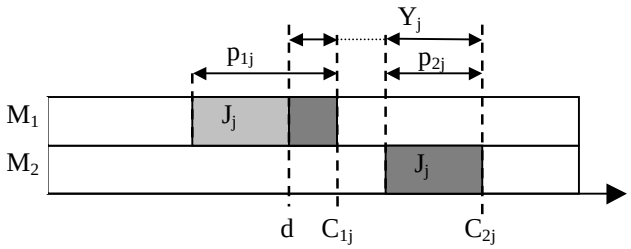

Figure 2.1. The late work parameter Y_j for job J_j in the two-machine flow shop environment

Problem $F2|d_j=d|Y_w$ stated above is binary NP-hard [15], since a polynomial transformation was constructed from the set partition problem to its decision counterpart and a dynamic programming method (DP) with pseudo-polynomial time complexity, $O(n^2d^4)$, was proposed [10]. This exact approach is important mainly from a theoretical point of view, because it determines the complexity status of the case under consideration. However, in practice, DP finds optimal solutions in reasonable time only for small problem instances. Thus, to solve the problem efficiently, heuristic approaches have to be proposed.

First, the list scheduling method was implemented for the problem under consideration and compared to the dynamic programming and enumerative algorithms [9, 11]. The list approach is a scheduling technique commonly used, especially for practical applications, because of its ease of implementation and the low time complexity. The computational experiments showed a very high efficiency of the list scheduling algorithm from the run time and the solution quality points of view. This heuristic constructed solutions with the criterion value of only 2.5% worse, on average, than the optimum. Taking into account these encouraging results, it was interesting, whether a further improvement of the solution quality could be obtained by extending the list scheduling approach with an additional search engine, i.e. by proposing metaheuristic methods.

3. Metaheuristic Approaches

In this paper, we present three metaheuristics (cf. e.g. [2, 14]) for problem $F2|d_j=d|Y_w$, i.e. simulated annealing (SA), tabu search (TS) and variable neighborhood search (VNS) methods. Similarly to the dynamic programming approach [10] and the list scheduling algorithm [9, 11], these approaches are based on the specific structure of an optimal solution of the problem under consideration. It was proven [10, 31], that in an optimal schedule, the set of early jobs, executed before a common due date, has to be sequenced in Johnson's order [23], which is optimal from the schedule length point of view. Thus, any method solving problem $F2|d_j=d|Y_w$ (cf. Fig. 3.1) has to select the first late job (L_1) in the system and to divide the remaining activities into two sets of early (E) and late jobs (L). Early jobs are scheduled by Johnson's algorithm, while the late activities are executed in non-increasing order of their weights w_j. In consequence, the crucial element of any method solving the problem is taking the decision whether a particular job is processed early or late in a schedule.

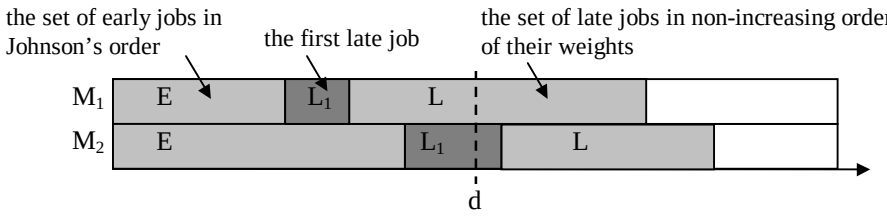

Figure 3.1. The general structure of an optimal solution for problem $F2|d_j=d|Y_w$ (with the first late job partially late on machine M_2)

The metaheuristic approaches investigated – SA, TS and VNS – are trajectory methods, which start their search from a certain initial solution and explore the solution space by moving from a current solution to a new one, picked up from its neighborhood, until the termination condition is met.

3.1. Initial Solution and Termination Condition

For all metaheuristic approaches proposed, the initial solution is determined either by Johnson's algorithm or by the list scheduling algorithm.

Johnson's method, designed for problem $F2||C_{max}$ [23], can be applied as a simple heuristic for the case under consideration, $F2|d_j=d|Y_w$. It orders jobs with a task on machine M_1 no longer than on M_2, i.e. with $p_{1j} \leq p_{2j}$, in non-decreasing order of p_{1j}, while the remaining jobs, with $p_{1j} > p_{2j}$, are sequenced in non-increasing order of p_{2j}. Because of the need of sorting jobs, the complexity of Johnson's method is bounded by O(nlogn).

On the contrary, the list algorithm is a constructive method, which builds a solution by scheduling jobs, selected according to a given priority dispatching rule [19], one by one on the machines. Based on the results obtained within the previous research [9,11], the maximum weight rule was applied, since it allowed constructing schedules of highest quality. The framework of this list procedure, which works in $O(n^2 \log n)$ time, is presented below [11].

Step 1: set $F = \emptyset$ and $A = J$, where F and A denote the set of executed and available jobs, respectively;
set $R = \emptyset$, where R denotes the set of feasible schedules constructed by the algorithm;
Step 2: if $A = \emptyset$ then go to Step 5
Step 3: take the job $\hat{J}_1$ from A with the maximum weight and set $A = A \setminus \{\hat{J}_1\}$;
set $F = F \cup \{\hat{J}_1\}$ and schedule F with Johnson's algorithm obtaining solution S;
if the schedule length of S exceeds d, then set $R = R \cup \{S\}$ assuming that the jobs in A are late;
if S is the first feasible solution found, then go to Step 4, otherwise go to Step 2;
Step 4: for each $J_x \in A$ schedule $F \setminus \{\hat{J}_1\} \cup \{J_x\}$ by Johnson's algorithm and execute jobs from $A \setminus \{J_x\} \cup \{\hat{J}_1\}$ late obtaining solution S;
if the schedule length of S exceeds d, then set $R = R \cup \{S\}$;
go to Step 2;
Step 5: select the best schedule from R to be the final solution and stop.

Since two methods of determining the initial solution are available, it is possible for the metaheuristics to start the solution space exploration from two different points, which may lead to final schedules of a different quality.

In the research reported, the search is terminated after exceeding a certain run time limit or after exceeding a given number of iterations without an improvement in the schedule quality.

3.2. Neighborhood Structures

As it was mentioned, a solution of problem $F2|d_j=d|Y_w$ is described as a sequence (E, L_1, L), where E denotes the set of early jobs scheduled in Johnson's order, L_1 is the first late job (executed partially late either on M_1 or on M_2) and L denotes the set of late jobs performed in non-increasing order of their weights. This feature of an optimal solution is the basis for the neighborhood structures applied within all three metaheuristics considered in this paper. In order to explore the solution space two move definitions were proposed involving a job shift (N_1) and jobs interchange (N_2).

According to the first move strategy, N_1, a new solution is generated by selecting a late job in a current schedule and shifting it to the set of early activities. Since all early jobs are scheduled in Johnson's order, it might happen that the position of the selected activity in the new Johnson's sequence does not ensure executing it totally early. In such a situation, it is necessary to move some early jobs preceding the chosen one in Johnson's sequence after the common due date d. On the other hand, after these modifications, some late jobs succeeding the selected activity in Johnson's order might need shifting before d to complete a schedule. The selection of particular activities may be done at random (selection rule S_1) or according to the weighted processing times of the jobs (selection rule S_2). This leads to two different neighborhood variants, which can be applied within metaheuristics. The idea of move N_1 is sketched below.

Step 1: move a job J_j selected from $L \cup \{L_1\}$ to the set E and calculate Johnson's schedule for E;
Step 2: if the job J_j is late in the new subschedule for E, then move to the set L some early jobs J_i selected in non-decreasing order of $(p_{1i}+p_{2i})w_i$ values or at random, until J_j becomes early or there are no more early jobs in E to be moved to L;
Step 3: add to the subschedule for E some jobs J_i from the set $L \cup \{L_1\}$ succeeding J_j in Johnson's order, selected in non-increasing order of $(p_{1i}+p_{2i})w_i$ values or at random, until the last job in E exceeds the common due date d becoming L_1;
Step 4: remove the job included to E as the last one and re-include it into L, then try all jobs from L as the first late job and select as L_1 the job for which the best criterion value has been obtained;
Step 5: schedule all late jobs J_i from the set L in non-increasing order of their w_i values.

Based on the definition of move N_2 provided below, a new schedule is obtained by choosing a pair of jobs - one from the set of late ones and one from the set of early ones - and interchanging them between these sets. The further solution modification is performed in a similar way as for move N_1.

Step 1: move a job J_j selected from the set E to the set L and disable it (i.e. exclude it from the further analysis, such that J_j cannot become early in Step 2);
Step 2: apply move N_1 to a job J_i selected from the set $L \cup \{L_1\}$.

The schedule modification for moves N_1 as well as N_2 requires the application of Johnson's algorithm, which takes O(nlogn) time. In consequence, both types of moves, enabling the exploration of the solution space, are performed in O(nlogn) time.

3.3. Simulated Annealing Method

The simulated annealing method proposed in the paper is based on the classical framework of this metaheuristic (cf. e.g. [2, 14, 24]), and it can be sketched as follows.

set the iteration number i to 0;
construct an initial schedule S_0;
set an initial temperature T_0 based on the total processing time of the jobs;
set S_0 to be the current solution S;
while termination conditions are not met do
    select at random a solution $\hat{S}$ from the neighborhood $N_k(S)$;
    calculate the change in the criterion value $\Delta Y_w = Y_w(\hat{S}) - Y_w(S)$;
    if $\Delta Y_w < 0$, then replace S by $\hat{S}$ else replace S by $\hat{S}$ with probability $P(\Delta Y_w, T_i)$;
    update temperature T_i according to the cooling scheme Q;
    set i=i+1.

The algorithm starts the search from an initial solution S_0 with an initial temperature T_0, whose value is set to the multiplied total processing time of the jobs (where the multiplication factor is a control parameter). At each iteration numbered with i, the method constructs a new solution $\hat{S}$ from a current one S according to the move definition N_1 or N_2, by shifting or interchanging jobs, respectively. This means that SA, performing move N_1 or N_2, picks at random one solution $\hat{S}$ from the neighborhood of the current solution S, $N_k(S)$, where $k \in \{1, 2\}$. The move configuration, i.e. the move type and the job selection rule within the move (cf. Section 3.2), are control parameters for the SA algorithm. If the new schedule improves the criterion value (i.e. if $Y_w(\hat{S}) - Y_w(S) < 0$, where $Y_w(s)$ denotes the criterion value for a schedule s), then it is accepted, otherwise it replaces the previous solution with a probability depending on the criterion value deterioration ΔY_w and the current temperature value at the i-th iteration T_i, i.e. with $P(\Delta Y_w, T_i)=\exp(-\Delta Y_w/T_i)$. At the end of each iteration, the temperature is decreased according to the geometrical reduction scheme Q, i.e. $T_{i+1}=\alpha T_i$, where $\alpha \in (0,1)$ is a control parameter. The preliminary computational experiments showed that the arithmetic reduction scheme ($T_{i+1}=T_i-\alpha$) is much less efficient for the problem under consideration, so it has been excluded from the further experimental analysis. SA finishes its search after reaching the termination condition mentioned in Section 3.1 or after decreasing the temperature to zero (more precisely speaking, to a value small enough).

3.4. Tabu Search Method

The tabu search method implemented for problem $F2|d_j=d|Y_w$ follows the classical scheme of this approach (cf. e.g. [2, 14, 16]) as it is sketched below.

construct an initial schedule S_0;
set S_0 to be the current solution S;
initialize tabu list T;
while termination conditions are not met do
    determine C as the neighborhood $N_k(S,\beta)$ restricted to the solutions without tabu status;
    replace S with the best solution $\hat{S}$ from C;
    update tabu list T.

In the TS algorithm, a new solution $\hat{S}$ is selected as the best not forbidden schedule from the neighborhood $N_k(S)$ generated for a current schedule S by applying move N_1 or N_2. The complete neighborhood $N_1(S)$ contains all schedules obtained by shifting every late job in S early, while the complete neighborhood $N_2(S)$ consists of solutions determined by exchanging every early job with every late job in S. However, it is possible to run the tabu search method with the restricted neighborhoods, $N_k(S, \beta)$, in which only β percent of jobs is considered for shifting or exchanging in particular move's definitions. Obviously, for β=100% the complete neighborhood is generated. In the restricted neighborhoods with β<100%, the jobs for shifting and exchanging are chosen at random. As we have mentioned, the best not forbidden schedule generated from the neighborhood becomes the starting point for the next iteration. In order to prevent the TS method from returning to the solutions already visited, the move leading to the schedule newly accepted is stored in the tabu list T. The attempt to reverse this move will cause the tabu status for related solutions in the neighborhoods which are considered in the following iterations. Consequently, only solutions without tabu status – constituting a candidate set C – are taken into account in the search process. The tabu list is managed according to the First In First Out rule (FIFO) and its length is determined with regard to the number of jobs (as the multiplied cardinality of the jobs set). The tabu search method stops after reaching the termination condition mentioned in Section 3.1 or when there is no solution in the neighborhood without the tabu status.

3.5. Variable Neighborhood Search Method

The variable neighborhood search method is a strategy using a local search algorithm and dynamically changing neighborhood structures (cf. e.g. [2, 14, 17]). The general idea of this approach applied within the presented research is given below.

construct an initial schedule S_0;
set S_0 to be the current solution S;
while termination conditions are not met do
    set k=1;
    while $k \leq 3$ do begin
        pick at random S' from the neighborhood $\check{N}_k(S)$;
        improve S' to S" with the TS or SA algorithm starting from a schedule S' as an initial solution;
        if $Y_w(S") < Y_w(S)$, then replace S with S" and set k = 1,
        else set k = k+1.

The VNS algorithm starts the search from an initial schedule S_0 as the current schedule S and it repeatedly applies three neighborhood definitions $\check{N}_k(S)$, k=1,2,3. Particular neighborhoods $\check{N}_1(S)$, $\check{N}_2(S)$, $\check{N}_3(S)$ have an increasing cardinality. i.e. $|\check{N}_1(S)|<|\check{N}_2(S)|<|\check{N}_3(S)|$. $\check{N}_1(S)$ is determined as the complete neighborhood $N_1(S)$ obtained by shifting all late jobs in S before the common due date (cf. Section 3.4), while $\check{N}_2(S)$ and $\check{N}_3(S)$ correspond to the neighborhood $N_2(S)$ restricted to schedules obtained by exchanging only β_2 and β_3 percent of early jobs and late jobs, where $\beta_2<\beta_3$. The values β_2 and β_3 are obviously control parameters of the VNS method.

At every iteration, VNS picks at random a solution S' from the neighborhood $\check{N}_k(S)$ generated from a current solution S. This selected schedule becomes the starting point for a local search method. The role of this local search procedure is played by the simulated annealing or the tabu search algorithm. That results in two versions of the variable neighborhood method: VNS-SA and VNS-TS, respectively. If a solution generated by the local search procedure, S'', improves the criterion value, then it is accepted and VNS returns to the first neighborhood definition. Otherwise, the local search is restarted from another neighbor solution generated according to the next neighborhood definition. The VNS algorithm finishes the solution space exploration after reaching the termination conditions mentioned in Section 3.1. Obviously, because the method calls SA or TS as a subprocedure, which performs certain number of its own iterations, the number of VNS iterations without an improvement, as the termination condition, has to be set to a rather small value.

4. Computational Experiments

Computational experiments performed within the reported research were devoted to comparing the efficiency of particular metaheuristic methods, as well as to evaluating the quality of their solutions with regard to optimal schedules. Moreover, the analysis of the test results made it possible to determine some specific features of the approaches proposed. The main computational experiments were obviously preceded by a careful tuning process [1, 20], during which we tested the metaheuristics efficiency for different values of control parameters in order to determine their best settings. The results of these preliminary experiments are given in Section 4.1. Then, in Section 4.2, the tuned metaheuristics are compared one to each other for large problem instances, as well as, in Section 4.3, to the exact methods for small problem instances.

4.1. Tuning of the Metaheuristic Algorithms

In the process of tuning all three metaheuristics, we used Johnson's algorithm to obtain the initial solution. Obviously, the list scheduling algorithm generated initial schedules of a higher quality than Johnson's one. Hence we used the list method in the main phase of the computational experiments, when the highest possible efficiency of the algorithms implemented was required. But during the tuning process, we wanted to check sensitivity of the methods to different settings of control parameters. Starting the search from a slightly worse solution (Johnson's one) we made some dependencies between control parameters values and the algorithms efficiency more visible.

4.1.1. Tuning of Simulated Annealing

In the tuning process for the simulated annealing method, we used a problem instance with 100 jobs, task processing times generated from the range [2, 40], job weights taken from the range [1, 10] and the common due date equal to 40% of the half of the total processing time of all jobs. As we have mentioned above, the initial solution was generated by Johnson's algorithm. Moreover, the preliminary tests showed that the arithmetic cooling scheme is much less efficient than the geometric one. Therefore, the following experiments were performed only for this latter (geometrical) cooling rule. The main objective of the tuning process was determining the control parameters setting that ensures the highest method efficiency in terms of the solution quality.

At the first phase of the tuning process, we used the time limit equal to 1 or 3 second(s), respectively, as the termination condition. For each of these termination values, 48 different control parameters settings were checked (that gave 96 tests in total). We used the initial temperature T_0 equal to the total processing time of the jobs multiplied by the factor 10, 20, 50, 100 and a cooling factor α equal to 0.9, 0.95, 0.999. For each pair of these parameters, we tested two possible moves N_1 (shifting a late job early) and N_2 (exchanging an early job with a late one) with two selection rules applied within them (the random rule S_1 and the rule based on the weighted processing times of the jobs S_2).

Based on the ranking of the solution quality, we observed that the selection rule S_2 involving the weighted processing times of the jobs always dominated the random rule S_1 ensuring a larger improvement of the criterion value (cf. Table 4.1.1.1). Moreover, the lower standard deviation value showed that the rule S_2 was more stable than S_1.

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
random selection rule S_1	0.00	3.33	8.87	2.61
weighted selection rule S_2	9.19	9.72	9.96	0.30

Table 4.1.1.1. The criterion value improvement in [%] with regard to the initial solution for the moves using the random (S_1) and the weighted processing time (S_2) selection rule for SA

Comparing two types of moves N_1 and N_2 (cf. Table 4.1.1.2), one could conclude that the move based on exchanging jobs (N_2) was less efficient than the move shifting a late job early (N_1). However, the smaller average criterion improvement obtained for N_2 for all tests was caused by the very poor quality of schedules generated only with the random rule S_1. Solutions constructed by interchanging two jobs, each of them selected at random, were much worse than schedules obtained by shifting a single activity, selected at random, early. For the weighted rule S_2, N_2 dominated N_1.

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
move N_1	0.00	6.26	9.96	4.16
move N_2	2.41	6.79	9.78	3.13

Table 4.1.1.2. The criterion value improvement in [%] with regard to the initial solution for the move shifting a job (N_1) and exchanging jobs (N_2) for SA

These observations were confirmed by the cross-comparison of the test results for particular move and selection rule settings (cf. Table 4.1.1.3). A current solution modification by exchanging jobs (N_2) selected according to their weighted processing times (S_2) guaranteed the highest criterion value improvement (for all possible settings of the remaining control parameters values, that is indicated by the zero standard deviation value).

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
all tests	0.00	6.53	9.96	3.69
N_2+S_2	9.96	9.96	9.96	0.00
N_1+S_2	9.19	9.47	9.78	0.24
N_2+S_1	0.00	2.56	7.88	2.68
N_1+S_1	2.41	4.11	8.87	2.29

Table 4.1.1.3. The criterion value improvement in [%] with regard to the initial solution for particular moves and selection rules configurations for SA

The analysis of the test results for the two termination conditions used in the experiments (i.e. 1- or 3-second time limit) showed rather weak influence of this termination value on the quality of the solutions (cf. Table 4.1.1.4). For the best configuration of the move and selection rule within the move discussed above, a longer computation time did not cause an additional solution improvement. For the worst move + rule configuration, the higher time limit ensured a slightly better method performance of the method than the lower time limit. However, in this case, SA was not able to obtain the solution quality ensured by the best move + rule configuration.

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
1-second limit	0.00	5.82	9.96	4.04
3-second limit	0.00	7.23	9.96	3.16

Table 4.1.1.4. The criterion value improvement in [%] with regard to the initial solution for the time limit equal to 1 or 3 seconds for SA

This analysis showed that the move type and the selection rule applied within the move influence the simulated annealing behavior the most. Hence, we additionally investigated the test results from this point of view to detect some additional features of the algorithm.

Looking at the time moment at which SA found the best solution (cf. Table 4.1.1.5), we saw that using the rule S_2, the method achieved the final solution faster than using S_1 (independently of the move type). As we could expect, the rule S_2 based on the instance description, i.e. on the weighted processing times of the jobs, appeared to be more efficient than the random rule S_1. It modified solutions in a systematic way leading the search to a local optimum more easily. In the experiments for S_2, the best solution was found quite early, after about 16% and 10%, respectively, of iterations, depending on the move type. It never happened that the method constructed the best schedule just before its termination (at most after about 67% of iterations). On the contrary, the random character of the rule S_1 spread the search within the solution space – the final schedule was generated after about 64% and 44%, respectively, of iterations, depending on the move type.

Sometimes, the best solution was determined even in the last step of SA (the maximum number of iterations to the best solution was equal to 100% of the total number of iterations). On average, the SA method found the best schedule quite fast, after about 34% of iterations, because initial solutions were rather close to the optimum. For this reason, the metaheuristic fell into a local optimum quite easily.

number of iterations to the best solution as [%] of the total number of iterations	minimum [%]	average [%]	maximum [%]	standard deviation [%]
all tests	0.16	33.94	100.00	39.38
N_2+S_2	0.67	16.68	67.10	20.68
N_1+S_2	0.16	10.49	46.02	16.41
N_2+S_1	0.65	64.43	99.49	41.61
N_1+S_1	2.52	44.18	100.00	43.49

Table 4.1.1.5. The number of iterations till constructing the best solution compared to the total number of iterations in [%] for particular move and selection rule configurations for SA

The temperature analysis confirmed the conclusions already formulated (cf. Table 4.1.1.6). In 87.5% of tests, the method using rule S_2 stopped because of cooling the system down. In the remaining cases, the final temperature was nearly equal to zero (≈0). SA with S_2, independently of the move variant, constructed a good solution at the early search steps, and it was usually terminated by the cooling mechanism, before exceeding the time limit. On the contrary, with the random selection rule S_1, the system was cooled down in only about 4% and 16%, respectively, of tests (depending on the move type). In the remaining tests, the method was stopped because of consuming all computation time allowed and the temperature decreased to about 30% of the initial value. Using the random rule, SA generated solutions of various quality and performed more time-consuming undirected search in the solution space, than applying the weighted processing time rule S_2. On average for nearly 49% of the tests the simulated annealing finished because of cooling the system down (i.e. as we have mentioned, mostly for the rule S_2).

final non-zero temperature as a [%] of the initial temperature	minimum [%]	average [%]	maximum [%]	standard deviation [%]	number of tests with zero final temperature [%]
all tests	≈0	27.47	92.22	39.36	48.96
N_2+S_2	≈0	≈0	≈0	≈0	87.50
N_1+S_2	≈0	≈0	≈0	≈0	87.50
N_2+S_1	≈0	29.03	92.22	39.57	4.17
N_1+S_1	≈0	33.92	90.21	41.51	16.67

Table 4.1.1.6. The percentage of tests for which SA stopped with zero temperature and the final temperature compared to the initial temperature value in [%] for tests for which SA stopped by reaching the termination condition for particular move and selection rule configurations (symbol "≈0" denotes 10^{-6} part of a percent)

The earlier research on problem $F2|d_j=d|Y_w$ showed that heuristic solutions constructed based on the special features of an optimal solution are very close to it in terms of the criterion value [10, 11]. This fact was confirmed by the quality of the solutions generated by SA in the tuning

process (cf. Table 4.1.1.7). Most solutions constructed by SA at particular iterations (about 78%) were worse than the current schedule. An initial solution quality close to the optimum really made the criterion value improvement very difficult and rare. For the S_2 rule, the initial schedule was improved at early iterations and a further decrease of the late work criterion was rather unlikely (about 95% of worse solutions). In the case of the random rule S_1, a current schedule was improved more often (in about 40% of iterations), because the method did not converge to a local optimum as fast as for S_2. It generated and accepted worse solutions and, then, improved them again, however, the final solution was usually worse than the one obtained with the S_2 rule.

number of worse solutions as [%] of the total number of analyzed solutions	minimum [%]	average [%]	maximum [%]	standard deviation [%]
all tests	49.56	77.75	99.93	19.53
N_2+S_2	75.86	95.61	99.91	7.09
N_1+S_2	75.45	95.04	99.93	7.90
N_2+S_1	49.67	59.07	75.48	7.54
N_1+S_1	49.56	61.26	83.05	10.83

Table 4.1.1.7. The percentage of iterations in which a solution worse from the current one was generated compared to the total number of analyzed solutions for particular move and selection rule configurations for SA

The higher convergence of SA applying the rule S_2 instead of S_1 was additionally confirmed by the number of worse solutions which were accepted during the search (cf. Table 4.1.1.8). For rule S_2, this number was equal to only about 6% of worse solutions. Because SA found the best schedule relatively early, the system cooled down fast, and the low temperature prevented it from accepting bad schedules. In the case of the random rule S_1, solutions of poor quality appeared very often at the beginning of the search, when the temperature and the probability of their acceptance were high (about 68% and 76%, respectively, of acceptances, depending on the move type).

number of accepted worse solutions as [%] of the number of analyzed worse solutions	minimum [%]	average [%]	maximum [%]	standard deviation [%]
all tests	0.06	39.53	100.00	39.95
N_2+S_2	0.70	6.31	34.62	9.73
N_1+S_2	0.06	6.93	37.59	11.67
N_2+S_1	24.80	76.45	99.38	25.73
N_1+S_1	12.93	68.42	100.00	33.55

Table 4.1.1.8. The percentage of iterations in which a solution worse than a current one was accepted compared to the total number of solutions worse than a current one or particular move and selection rule configurations in SA

In consequence of the fast convergence to a local optimum observed for SA with the rule S_2, its average iteration time was much shorter than for the rule S_1 (cf. Table 4.1.1.9). SA with S_2 generated a good solution at the beginning of the search. Because schedules constructed in the later steps were usually worse than the current one (cf. Table 4.1.1.7) and the criterion value deterioration made their acceptance very unlikely (cf. Table 4.1.1.8) most of them were rejected. As we have already mentioned, for the random rule S_1, the search process was less stable and more

time consuming. Hence, SA with S_2 required only about 50 microseconds per iteration, while SA with S_1 consumed about 15 milliseconds. The difference in the computational times between S_1 and S_2 results mainly from the implementation issues. In the implementation used, accepting a new solution (more often for S_1) is much more time consuming than rejecting it (more often for S_2). Moreover, selecting jobs according to the weighted processing time rule (S_2) is supported by some specialized fast data structures, which could not be applied for the random selection (S_1).

time per iteration	minimum [μsec.]	average [μsec.]	maximum [μsec.]	standard deviation [μsec.]
all tests	51	6 891	22 074	7 262
N_2+S_2	53	55	61	2
N_1+S_2	51	56	65	3
N_2+S_1	10 113	15 293	22 074	3 477
N_1+S_1	8 372	12 160	21 255	2 660

Table 4.1.1.9. The time in [μsec.] spent for a single iteration of SA for particular move and selection rule configurations

Taking into account the very vague differences among particular control parameters settings, we performed additional computational experiments for the best move configuration, i.e. modifying a solution by exchanging jobs based on the weighted processing time (N_2+S_2). We tested the following values of parameters:

- the initial temperature 10, 20, 50 and 100 times larger than the total processing time of the jobs,
- the cooling factor equal to 0.9, 0.933, 0.966 and 0.999,
- the termination condition as the number of iterations without an improvement equal to 1, 2, 5, 10, 20 and 50 times the number of jobs.

That gave 96 experiments in total, whose results are summarized in Table 4.1.1.10.

all test	minimum	average	maximum	standard deviation
criterion improvement [%]	1.12	8.57	9.96	2.69
time [sec.]	0.01	0.15	0.89	0.17
iterations to the best solution [%]	7.05	49.27	87.23	22.62
worse vs. all solutions	50.42	82.64	99.20	18.41
worse accepted vs. all worse solutions	1.06	29.82	100	40.23
time per iteration [μsec.]	50	56	70	3.91

Table 4.1.1.10. Test results of additional tuning experiments for the move N_2 with the selection rule S_2 for SA

In the additional tests, the simulated annealing continued its search until the given number of iterations without an improvement was exceeded. The average run time equal to 0.15 sec. showed that the 3-second time limit used in the previous stage of the experiments was sufficiently long. Moreover, because in 69.8% of tests, the system stopped with zero temperature. This means that the new termination condition used was large enough to cool the system down. Changing the termination condition from the time limit to the number of iterations without an improvement caused that the number of iterations to the best solution increased from 17% to about 50% of the total iteration number (cf. Table. 4.1.1.5). In the case of a time limit, the total number of iterations

was larger because the method continued the search without an improving the solution, till consuming the whole computation time assigned to it.

Because the results of these additional experiments were still very similar, in order to determine the best control parameters setting for SA, we restricted the analysis first to the configurations ensuring the highest solution quality improvement equal to 9.96% (48 configurations from 96), then to the ones with zero final temperature (47 configurations from 48) and, finally, to the settings for which the run time did not exceed 0.1 sec. (12 configurations from 47). Analyzing these 12 best configurations, given in Table 4.1.1.11, we saw that there are only five dominant configurations with respect to the distinct values of the initial temperature T_0 factor (expressed as the multiplication of the total processing time of the jobs) and the cooling factor. We decided to test them additionally with the termination factor equal to the doubled number of jobs. These configurations are marked with an asterisk in Table 4.1.1.11. Moreover, we completed this set with the configuration with a termination factor equal to 2, T_0 factor set to 10 and a cooling factor fixed as 0.999 to check the influence of a large cooling factor value on the solution quality, too.

	termination factor	T_0 factor	cooling factor	run time [sec.]	number of iterations to the best solution [%]	number of worse solutions vs. total number of analyzed solutions [%]
	10	100	0.9	0.080	32.57	95.01
	5	100	0.9	0.053	49.14	92.47
*	**2**	**100**	**0.9**	0.037	70.72	89.17
	1	100	0.9	0.039	82.85	87.31
	10	50	0.9	0.076	27.48	95.65
	5	50	0.9	0.049	43.12	93.17
*	**2**	**50**	**0.9**	0.033	65.46	89.64
*	**5**	**20**	**0.966**	0.094	71.28	92.42
*	**5**	**20**	**0.933**	0.080	67.36	95.23
	10	10	0.966	0.084	35.98	93.53
	5	10	0.966	0.059	52.92	90.49
*	**2**	**10**	**0.966**	0.043	73.75	86.75

Table 4.1.1.11. Test results for 12 top control parameters settings ensuring the highest solution improvement of 9.96 % within 0.1 sec. for which the zero final temperature was reached for SA

In the final experiment, we used 10 instances of the problem under consideration, i.e. two instances for each number of jobs equal to 10, 20, 50, 60, 75. They were solved by SA with 6 different control parameters settings, carefully selected in the previous stage of the analysis. That gave 60 tests in total.

The ranking of these control parameters settings with regard to the average criterion value improvement is presented in Table 4.1.1.12, where particular configurations are characterized with the termination factor (the number of iterations without an improvement as the multiplied number of jobs), the initial temperature factor (the initial temperature as a multiplied total processing time of the jobs) and the cooling factor.

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
all tests	1.30	10.78	18.52	4.31
5-20-0.933	4.75	11.99	18.18	3.75
5-20-0.966	4.75	11.78	18.52	4.05
2-50-0.9	1.30	11.18	17.96	4.47
2-100-0.9	1.30	11.06	17.89	4.72
2-10-0.966	1.30	10.54	17.89	4.59
2-10-0.999	1.30	8.16	11.82	2.88

Table 4.1.1.12. The criterion value improvement in [%] for selected control parameters settings for SA

The largest average criterion value improvement was obtained for the configuration 5-20-0,933, which ensured additionally almost the smallest standard deviation. Because of the high value of the termination condition the run time was longer than for other control parameters settings, but the differences were small (cf. Table 4.1.1.13).

run time	minimum [sec.]	average [sec.]	maximum [sec.]	standard deviation [sec.]
all tests	0.000	0.012	0.049	0.012
2-10-0.999	0.000	0.005	0.012	0.004
2-10-0.966	0.000	0.010	0.022	0.008
2-50-0.9	0.000	0.010	0.025	0.008
2-100-0.9	0.000	0.013	0.030	0.011
5-20-0.933	0.001	0.017	0.049	0.015
5-20-0.966	0.001	0.019	0.048	0.016

Table 4.1.1.13. The run time in [sec.] for selected control parameters settings for SA

Based on a careful analysis of all computational experiments, we selected the following control parameters values as the result of the tuning process for the SA method:

- the solution modification based on the move exchanging jobs selected with the weighted processing time rule: N_2+S_2,
- the termination condition set as the number of iterations without an improvement 5 times as large as the number of jobs,
- the initial temperature 20 times as large as the total processing time of the jobs,
- the cooling scheme with the geometrical scheme with a cooling factor equal to 0.933.

4.1.2. Tuning of Tabu Search

The tabu search method is a trajectory method constructing a set of neighborhood solutions at each iteration instead of generating a single solution as the simulated annealing approach does. Hence, taking into account the longer expected run time of TS, we used an instance with 30 jobs, task processing times generated from the range [2, 40], job weights taken from the range [1, 5] and the common due date equal to 40% of the half of the total processing time of the jobs. Similarly as for SA, we performed the tuning process using Johnson's solution as the initial one and with the

run time limit set to 1 or 3 seconds. Moreover, the following control parameters settings were investigated:

- the tabu list length equal to 30%, 60% and 90% of the number of jobs,
- the neighborhood based on the job shifting (N_1) or jobs exchanging (N_2) with the random selection rule (S_1) or the rule based on the weighted processing times of the jobs (S_2),
- the neighborhood size (β) restricted to solutions obtained by shifting/exchanging 33%, 67% or 100% of the number of candidate jobs.

That gave 72 test sets in total.

In the case of the tabu search method the dominance of the selection rule S_2 (based on the weighted processing times of the jobs) over the random rule S_1 was not so visible as for the simulated annealing algorithm (cf. Table 4.1.2.1 vs. Table 4.1.1.1). Nevertheless, the selection rule based on the instance description S_2 ensured a slightly higher solution quality than the random one. Moreover, S_2 was much more stable than S_1 in terms of the standard deviation. These results disclose the differences between TS and SA. Since SA constructed a single solution at each iteration, the quality of the current solution influenced significantly the quality of the final one. The schedule modification based on the instance features (i.e. on the weighted processing times of the jobs) resulted in a better current solution and, consequently, in a better schedule being the result of the whole search process. In the case of TS, a whole set of neighborhood solutions was constructed at a single iteration. Therefore, the procedure according to which jobs are selected to build new schedules did not influence the quality of a new current solution as much as one could observe for the SA method. Taking into account the fact, that the initial solution of the problem under consideration was not far from the optimum, different move concepts resulted in similar neighborhoods. Thus, the efficiency of particular selection rules S_1 and S_2 applied within TS appeared to be nearly alike.

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
random selection rule S_1	9.27	14.29	15.88	2.02
weighted selection rule S_2	14.76	15.61	16.17	0.59

Table 4.1.2.1. The criterion value improvement in [%] with regard to the initial solution for the neighborhoods with the random (S_1) and weighted processing time (S_2) selection rule for TS

Similar conclusions were drawn from the comparison of the two neighborhood concepts proposed. The move exchanging jobs, N_2, resulted in only slightly better schedules than the move shifting a selected job early, N_1 (cf. Table 4.1.2.2).

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
move N_1	10,76	14,51	15,88	1,49
move N_2	9,27	15,39	16,17	1,63

Table 4.1.2.2. The criterion value improvement in [%] with regard to the initial solution for the neighborhood shifting a job (N_1) and exchanging jobs (N_2) for TS

As for the simulated annealing approach, the comparison of particular combinations of the move and the selection rule (cf. Table 4.1.2.3) showed that exchanging jobs selected according to their weighted processing times (N_2+S_2) always ensured the highest solution quality (independently on other control parameter settings of the algorithm which is indicated by the zero standard deviation).

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
all tests	9.27	14.95	16.17	1.63
N_2+S_2	16.17	16.17	16.17	0.00
N_1+S_2	14.76	15.04	15.36	0.26
N_2+S_1	9.27	14.61	15.88	2.03
N_1+S_1	10.76	13.97	15.88	1.95

Table 4.1.2.3. The criterion value improvement in [%] with regard to the initial solution for particular neighborhood and selection rule configurations for TS

Furthermore, similarly as for SA, the influence of the time limit on the solution quality was not significant (cf. Table 4.1.2.4). TS with N_2 and S_2, which appeared to be the best neighborhood configuration, generated exactly the same results within the 1- and 3-second time limits. For worse control parameters settings, the longer run time resulted in a slightly higher criterion value improvement.

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
1-second limit	9.27	14.67	16.17	1.86
3-second limit	10.76	15.23	16.17	1.29

Table 4.1.2.4. The criterion value improvement in [%] with regard to the initial solution for time limits equal to 1 or 3 seconds for TS

Comparing the percentage of the number of iteration till the moment when the best solution was determined (cf. Table 4.1.2.5), we noticed that for the neighborhood N_2+S_2, a final solution was found very fast, i.e. at the beginning of the search. The move exchanging early and late jobs taking into account their weighted processing times constructed good solutions, which could be hardly improved in the following iterations. For the remaining move + rule configurations reaching the local optimum was more difficult and the best solutions were found in the later iterations of the search process.

number of iterations to the best solution as [%] of the total number of iterations	minimum [%]	average [%]	maximum [%]	standard deviation [%]
all tests	0.36	57.05	100.00	38.25
N_2+S_2	0.36	1.08	3.10	0.80
N_1+S_2	40.43	73.60	100.00	21.97
N_2+S_1	80.00	95.74	100.00	8.00
N_1+S_1	22.41	57.77	81.63	20.04

Table 4.1.2.5. The number of iterations till the moment of constructing the best solution compared to the total number of iterations in [%] for particular neighborhood and selection rule configurations for TS

The influence of the move type applied within the procedure constructing a neighborhood on the search process was better visible when the number of solutions analyzed per single iteration was investigated (cf. Table 4.1.2.6). Exchanging jobs, independently of the rule selecting those jobs, lead to a larger number of distinct neighbor solutions than moving some late activities early. It is obvious that there were at least as many pairs of early and late jobs to be exchanged (N_2) as late jobs to be moved early (N_1).

the number of analyzed solutions per iteration	minimum [No]	average [No]	maximum [No]	standard deviation [No]
all tests	2	13	30	9
N_2+S_2	9	19	30	9
N_1+S_2	3	7	11	3
N_2+S_1	8	19	30	9
N_1+S_1	2	5	9	3

Table 4.1.2.6. The number of analyzed solutions per iteration for particular neighborhood and selection rule configurations for TS

Similarly as for the SA method, we performed additional computational experiments for the best neighborhood configuration (N_2+S_2) with:

- the tabu list length equal to 30%, 60%, 90%, 300%, 600% and 900% of the number of jobs,
- the neighborhood size restricted to solutions obtained by exchanging 33%, 67%, 100% of the total number of candidate jobs,
- the termination condition set as the number of iterations without an improvement equal to the number of jobs as well as 2, 5, 10 times as large as the number of jobs.

That gave 72 experiments in total.

all test	minimum	average	maximum	standard deviation
criterion improvement [%]	15.58	16.11	16.17	0.19
time [sec.]	0.012	0.199	0.916	0.229
iterations to the best solution [%]	1.32	14.97	51.22	13.48
number of iterations [No]	33	158	433	120
solutions per iteration [No]	8	19	30	9
time per iteration [msec.]	0	1405	3013	1030

Table 4.1.2.7. Test results for additional tuning experiments for the neighborhood N_2 with the selection rule S_2 for TS

In the additional tests (cf. Table 4.1.2.7), the termination condition was determined by the number of iterations without an improvement. Since the average run time was equal to 0.199 sec., we could state that the 3-second time limit used in the previous experiments was large enough similarly as for the SA algorithm. However, the results of the additional tests were even more alike for the tabu search than for the simulated annealing approach. 63 control parameter settings among 72 resulted in the same solution quality, and for 32 among these 63 configurations the run time did not exceed 0.1 sec. Therefore, the choice of 5 control parameters out of 32 settings same as above, used in the further tuning process, was done in a more arbitrary way than for SA (cf. Table 4.1.2.8).

Among the configurations with the shortest and longest termination condition (the termination factor equal to 1 and 10 times as large as the number of jobs), we chose those ones which had the shortest run time (Settings 1 and 5 in Table 4.1.2.8). For the termination condition set as the number of iterations without an improvement 5 times as large as the number of jobs, all control parameters settings in the restricted set of 32 top results mentioned above contained the neighborhood generated for 33% of the number of candidate jobs. We decided arbitrarily to combine these two parameters (the termination condition and the neighborhood size) with a medium tabu list length equal to 90% of the number of jobs (Setting 4 in Table 4.1.2.8). Finally, for the termination condition equal to the number of iterations without an improvement set as the doubled number of jobs, the top ranked control parameters settings contained only two types of neighborhoods: generated for 33% and 67% of the number of candidate jobs. We combined these parameters with two different tabu list lengths obtaining Settings 2 and 3 in Table 4.1.2.8.

setting	termination factor	tabu list length	neighbor-hood size	time [sec.]	number of iterations to the best solution [%]	number of iterations	number of solutions per iterations
1	1	900	67	0.049	25.00	39	19
2	2	30	67	0.033	14.29	69	19
3	2	300	33	0.083	51.22	122	9
4	5	90	33	0.054	35.34	231	9
5	10	300	33	0.084	17.36	362	9

Table 4.1.2.8. Test results for 5 top control parameters settings selected for additional experiments for TS

Similarly as in the SA tuning process, we compared the efficiency of these 5 selected sets of the control parameters values for 10 different problem instances (two instances per 10, 25, 30, 42 and 50 jobs) which gave 50 tests in total. The results of these additional experiments in terms of the criterion value improvement (cf. Table 4.1.2.9) and the time efficiency (cf. Table 4.1.2.10) confirmed that the TS method was rather insensitive to the control parameters settings for the problem under consideration.

criterion value improvement	minimum [%]	average [%]	maximum [%]	standard deviation [%]
all tests	0.00	14.10	40.66	9.34
1-900-67 (Setting 1)	6.24	14.99	40.66	9.68
2-30-67 (Setting 2)	6.24	14.99	40.66	9.68
10-300-33 (Setting 5)	0.00	13.52	34.44	9.05
2-300-33 (Setting 3)	0.00	13.50	34.44	9.05
5-90-33 (Setting 4)	0.00	13.50	34.44	9.05

Table 4.1.2.9. The criterion value improvement in [%] with regard to the initial solution for selected control parameters settings for TS

Since the differences in the tabu search method performance for the analyzed control parameters settings were very small and the tabu search approach was, in general, more time consuming than the simulated annealing one, we took for the further computational experiments the configuration which required the shortest run time (i.e. Setting 3).

run time		minimum [sec.]	average [sec.]	maximum [sec.]	standard deviation [sec.]
all tests		0.000	0.136	0.707	0.184
2-300-33	(Setting 3)	0.000	0.051	0.158	0.057
5-90-33	(Setting 4)	0.000	0.102	0.332	0.119
1-900-67	(Setting 1)	0.001	0.122	0.378	0.137
10-300-33	(Setting 5)	0.000	0.198	0.622	0.224
2-30-67	(Setting 2)	0.001	0.210	0.707	0.256

Table 4.1.2.10. The run time in [sec.] for selected control parameters settings for TS

Taking into account all computational experiments, we selected the following control parameters values as the result of the tuning process for the TS method:

- the neighborhood generated by exchanging jobs (N_2) based on the weighted processing times of the jobs (S_2),
- the neighborhood generated from 33% of candidate jobs,
- the termination condition set as the number of iterations without an improvement equal to the doubled number of jobs,
- the tabu list length equal to 300% of the number of jobs.

It is worth to be mentioned, that the rather large tabu list length chosen resulted from the specificity of the problem under consideration. Taking into account the fact that the initial solution was nearly optimal, we could not allow the method to return to a solution already visited, if we wanted to force it to get closer to the optimum.

4.1.3. Variable Neighborhood Search Method Tuning

In the preliminary experiments for the variable neighborhood method, we used the simulated annealing or the tabu search as a local search procedure, using the control parameters settings determined in the tuning process reported in Section 4.1.1 and Section 4.1.2, respectively. We tested VNS-SA for a problem instance with 500 jobs and VNS-TS for an instance with 30 jobs. The task processing times were generated from the range [2, 40], the job weights were taken from the range [1, 10] for VNS-SA and from the range [1, 5] for VNS-TS. Finally, the due date was set to 40% of the half of the total processing time of the jobs.

Actually, there are only two parameters controlling the behavior of the VNS method implemented: the termination condition and the neighborhood configuration.

The termination condition decides how many times the main loop of the algorithm is performed and it does not influence the termination condition of an embedded local search procedure. Taking into account the good quality of the initial solution and the high efficiency of SA and TS used as a subroutine, enforcing long-lasting runs of VNS seems to be pointless. Therefore, during the tuning process, the VNS termination condition was settled to only 1, 3 or 5 iterations without an improvement.

The VNS method is based on the same neighborhood definitions, which are applied within SA and TS. In order to lead the search from a narrower to a wider neighborhood, VNS starts with

the neighborhood shifting a job (N_1) and, then, it passes to two neighborhoods exchanging jobs (N_2). The size of these neighborhoods is controlled by the size of the set of jobs being candidates for shifting/exchanging. We decided to check two configurations denoted by 100/20/50 and 100/50/100. For example, the setting 100/20/50 denotes that the first neighborhood (based on the move N_1) was generated for all candidate jobs (100%), the second neighborhood (based on the move N_2) exchanged 20% of the number of candidate jobs, and the third neighborhood (based on the move N_2 too) exchanged 50% of the number of candidate jobs.

Combining both control parameters values, the termination condition and the neighborhood definition, we obtained 6 control parameters settings for the computational verification.

control parameters settings	VNS-SA		VNS-TS	
	criterion value improvement [%]	run time [sec.]	criterion value improvement [%]	run time [sec.]
1-100/20/50	11.3	3.401	16.2	0.093
1-100/50/100	11.3	3.436	16.2	0.094
2-100/20/50	11.3	4.799	16.2	0.239
2-100/50/100	11.3	4.803	16.2	0.240
3-100/20/50	11.3	6.187	16.2	0.389
3-100/50/100	11.3	6.184	16.2	0.505
minimum	11.3	3.40	16.2	0.093
average	11.3	4.80	16.2	0.260
maximum	11.3	6.19	16.2	0.505
standard deviation	0.0	1.13	0.0	0.149

Table 4.1.3.1. The criterion value improvement in [%] with regard to the initial solution and the run time in [sec.] for VNS method with SA and TS algorithms for different control parameters settings (i.e. the termination condition and neighborhood definitions)

The preliminary tests showed that the VNS method was insensitive to the control parameters settings (cf. Table 4.1.3.1). VNS with SA as well as with TS found the best solution in its first iteration, that means after the first run of the embedded local search method. In consequence, particular sets of control parameters could not be distinguished from the efficiency point of view. The time efficiency obviously reflected the termination condition – the larger the number of iterations without an improvement, the longer run time of the method.

Taking into account the fact that the initial solution was close to the optimum as well as the efficiency of the TS and SA method, a further solution improvement by the VNS algorithm was very difficult to be obtained. It is rather unlikely to found significantly better schedules by restarting SA or TS from different initial points in this case. Thus, in the main experiments, we decided to use the control parameters setting ensuring the highest degree of freedom for the VNS method, i.e.:

- the termination condition set to 3 iterations without an improvement,
- the neighborhood configuration with the candidate sets restricted to 100%, 50%, 100% of the number of candidate jobs.

4.2. Metaheuristic Methods Comparison

In order to compare the efficiency of particular metaheuristics one to each other as well as to the list scheduling approach, we analyzed 20 instances of the problem under consideration with:

- the number of jobs equal to 20 and 200 (two instances per each number of jobs), as well as 50, 80, 110 and 140 (four instances for each number of jobs),
- the task processing times generated from the range [2, 40],
- the job weights taken from the range [1, 5],
- the due date equal to 40% of the half of the total processing time of the jobs.

For each test instance, particular metaheuristic methods were run four times: starting from Johnson's schedule or from the list schedule as the initial solution, and with a limited number of iterations without an improvement or with a time limit as the termination condition. The remaining control parameter settings were determined during the tuning process. That gave 80 tests in total.

In general, taking into account the best results of particular metaheuristics among the 4 runs mentioned above (cf. Table 4.2.1 and Table 4.2.2), these methods improved the list solution in 70% of the tests (14 instances among 20) with 2.82‰ on average. The smallest improvement was equal to 0.2‰, while the maximum one equals 6.8‰. Taking into account the fact that the list algorithm generates schedules of a very high quality, i.e. a nearly optimal one (cf. [11]), the further improvement achieved by metaheuristic methods should be regarded as a considerable one, especially, taking into account the rather simple structure of these methods and their short run time. The computational experiments showed that even a very good schedule can be still improved without a huge time effort, by performing a systematic search in the solution space.

	distance to the best result in [‰]	LA	SA(J)	SA(L)	TS(J)	TS(L)	VNS-SA(J)	VNS-SA(L)	VNS-TS(J)	VNS-TS(L)
No of iterations	minimum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	average	1.98	0.01	0.05	3.69	1.18	0.03	0.00	3.40	1.16
	maximum	6.80	0.15	0.74	31.72	6.80	0.70	0.00	31.72	6.80
	standard deviation	2.04	0.03	0.17	7.72	1.87	0.15	0.00	7.54	1.89
	No of tests among 20 with the best result	6	19	18	10	10	19	20	10	11
5-second time limit	minimum	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	average	1.98	0.00	0.00	3.67	1.21	0.00	0.00	3.34	1.16
	maximum	6.80	0.00	0.00	31.72	6.80	0.00	0.00	27.59	6.80
	standard deviation	2.04	0.00	0.00	7.73	1.86	0.00	0.00	6.73	1.89
	No of tests among 20 with the best result	6	20	20	10	9	20	20	9	11

Table 4.2.1. The distance to the best criterion value for particular instances (for 20 tests) for the list algorithm (LA) and the metaheuristics starting from Johnson's schedule (J) or a list schedule (L) with the number of iterations without an improvement and a 5-second time limit as the termination condition

The following detailed analysis of the computational experiments disclosed some interesting features of the metaheuristics implemented within the presented research (Table 4.2.1, Figure 4.2.1 and Figure 4.2.2.).

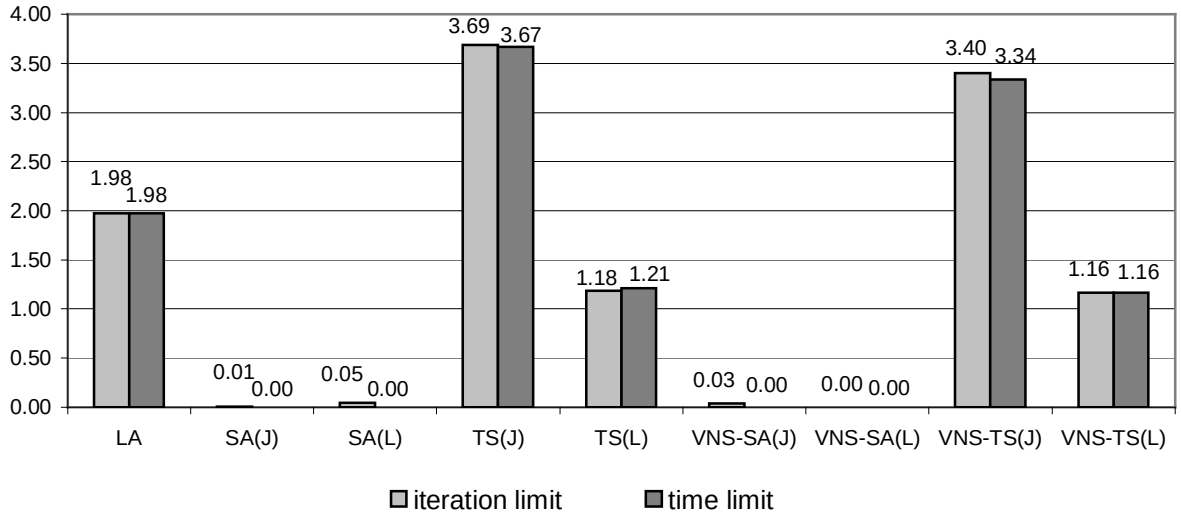

Figure 4.2.1. The average distance to the best solution in [‰] for the list algorithm (LA) and particular metaheuristics (for 20 tests) starting from Johnson's schedule (J) or a list schedule (L) for the number of iterations without an improvement and a 5-second time limit as the termination condition

Comparing particular methods, one could notice the superiority of the simulated annealing one. The SA algorithm constructed the best schedule for all instances analyzed, when the termination condition was set to a 5-second limit (Figure 4.2.2). In this case, restarting SA from different initial solutions by the variable neighborhood algorithm did not result in an improvement of the solution quality, because of the high quality of the schedule found in the first run of SA. When the number of iterations without an improvement was applied as the termination condition, SA found the best solution for only 2 instances less with a list schedule as the initial solution and for only 1 instance less than starting from Johnson's schedule. In both cases SA significantly dominated the tabu search approach (Figure 4.2.1). Since the simulated annealing algorithm was extremely fast, a 5-second time limit enforced this method to explore the solution space in time 2 and even 3 factors longer than the time resulting from the termination condition based on the number of iterations without an improvement (cf. Table 4.2.4 with a 5-second time limit). The long run time limit allowed SA to leave the local optimum within the number of iterations which would be not possible for the latter termination condition. These results showed that forcing a fast metaheuristic to a long-lasting search might cause an additional criterion value improvement. In general, a similar effect could be obtained by restarting a method from different initial solutions, as it was performed within the variable neighborhood search algorithm. However, the computational experiments showed that the efficiency of the local search procedure was crucial for the efficiency of VNS for the problem under consideration. This method rarely improved the solution constructed by SA at the first VNS iteration.

The tabu search algorithm constructed solutions worse than those found by SA (Figure 4.2.1). TS achieved the best criterion value for only 9 and 10, respectively, instances (depending on the control parameters values) improving a list schedule for 3 and 4 instances from the test set (Figure 4.2.2). Embedding TS within VNS sometimes made it possible to obtain better results than in a single run of TS (for at most 2 instances more, when the method started from a list schedule

and worked within a 5-second time limit). It is interesting, that for one test instance VNS-TS generated a schedule worse than TS. Within VNS, TS started its search from a solution taken at random from the neighborhood of the initial solution, which might be worse than this initial one. For this reason, a single run of TS starting from a good schedule, could result in a better solution than a few runs of TS within VNS initialized with a worse schedule.

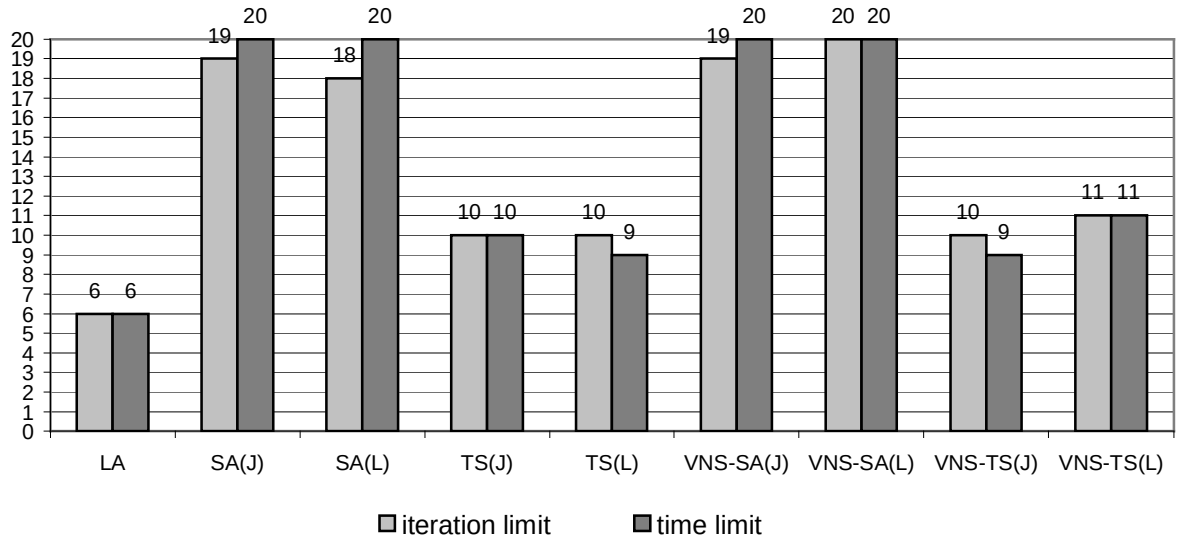

Figure 4.2.2. The number of test instances in which particular methods constructed a solution with best criterion value (for 20 tests) starting from Johnson's schedule (J) or a list schedule (L) for the number of iterations without an improvement and a 5-second time limit as the termination condition

Summing up, the variable neighborhood search method with simulated annealing as a local search procedure and the list schedule as the initial solution (VNS_SA(L)) appeared to be the best choice for the problem under consideration. It constructed the best schedule for all test instances independently of the termination condition applied. The superiority of VNS-SA resulted from the high efficiency of the simulated annealing algorithm. It was possible to achieve the same solution quality level, by extending the SA run time limit to 5 seconds or by restarting SA from different initial solutions within VNS. The tabu search approach appeared to be much less efficient, and introducing a time limit as the termination condition as well as multiple restarting TS within VNS did not result in a significant solution quality improvement. Taking into account the fact that the initial schedule was close to the optimum, searching the solution space by a single modification of a current schedule (as in SA) appeared to be a better approach than generating the whole neighborhood containing schedules mostly worse than the current one (as in TS). Moreover, the computational experiments showed that forcing a long run time limit, longer than the time determined by the tuned number of iterations without an improvement as the termination condition, one can increase the final solution quality.

However, the efficiency of the method depends not only on the termination condition but also on the quality of the initial schedule (cf. Table 4.2.1, Figure 4.2.1). TS and TS-VNS generated much better schedules starting from the list solution than from Johnson's one with a worse criterion value. Because SA and VNS-SA, in general, generated the best schedules for particular instances,

this dependence between the quality of the final solution and the initial schedule cannot be observed. But, looking at the number of the best results obtained (cf. Table 4.2.1, Figure 4.2.2), one could draw a similar observation than for TS and VNS-TS. Summing up, starting the search from the better schedule - the list solution - the metaheuristics were usually able to obtain a higher final solution quality than in the case of starting from Johnson's schedule.

In order to compare the performance of the metaheuristic methods implemented more deeply, we restricted the analysis to only those instances for which a particular method did not find the best solution (Table 4.2.2, Figure 4.2.3.). This made some observations stated above more apparent.

	distance to the best result in [‰]	LA	SA(J)	SA(L)	TS(J)	TS(L)	VNS-SA(J)	VNS-SA(L)	VNS-TS(J)	VNS-TS(L)
No of iterations	minimum	0.20	0.15	0.20	0.15	0.20	0.70	0.00	0.38	0.20
	average	2.82	0.15	0.47	7.37	2.37	0.70	0.00	6.81	2.59
	maximum	6.80	0.15	0.74	31.72	6.80	0.70	0.00	31.72	6.80
	standard deviation	1.89	0.00	0.27	9.59	2.05	0.00	0.00	0.95	0.21
	No of tests among 20 without the best result	14	1	2	10	10	1	0	10	9
5-second time limit	minimum	0.20	0.00	0.00	0.28	0.20	0.00	0.00	0.28	0.20
	average	2.82	0.00	0.00	7.34	2.21	0.00	0.00	6.06	2.59
	maximum	6.80	0.00	0.00	31.72	6.80	0.00	0.00	27.59	6.80
	standard deviation	1.89	0.00	0.00	9.61	2.03	0.00	0.00	8.11	2.05
	No of tests among 20 without the best result	14	0	0	10	11	0	0	11	9

Table 4.2.2. The distance to the best criterion value for particular instances for the list algorithm (LA) and the metaheuristics starting from Johnson's schedule (J) or a list schedule (L) for two termination conditions restricted to those instances for which a particular method did not find the best solution

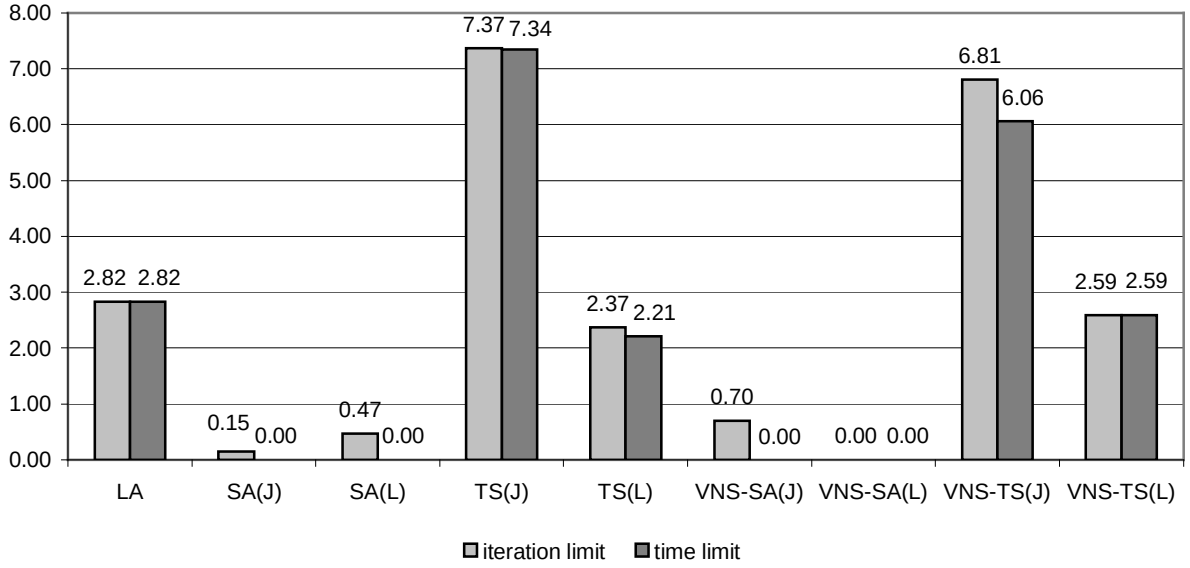

Figure 4.2.3. The average distance to the best solution in [‰] for the list algorithm (LA) and particular metaheuristics starting from Johnson's schedule (J) or a list schedule (L) for two termination conditions restricted to those instances for which a particular method did not find the best solution

SA and VNS-SA were not the best metaheuristic only for 1 or 2 instances depending on the control parameters values, for these tests TS or VNS-TS were more efficient. However, in the cases mentioned, the quality of SA and VNS-SA solutions differed from the best result of only fractions of per mill. On the contrary, TS and VNS-TS were less efficient than SA and VNS-SA for about 10 instances, generating solutions of about 7‰ and 2.5‰ worse starting from Johnson's and the list schedule, respectively.

	difference in solutions quality in [‰]	SA(J)	SA(L)	TS(J)	TS(L)	VNS-SA(J)	VNS-SA(L)	VNS-TS(J)	VNS-TS(L)
iteration limit dominating time limit	minimum	-	-	0.149	-	-	-	0.753	-
	average	-	-	0.214	0.557	-	-	1.467	-
	maximum	-	-	0.279	-	-	-	2.179	-
	No of tests	0	0	2	1	0	0	2	0
time limit dominating iteration limit	minimum	-	0.196	-	-	-	-	0.139	-
	average	0.149	0.471	0.753	-	-	0.697	2.206	-
	maximum	-	0.745	-	-	-	-	4.274	-
	No of tests	1	2	1	0	0	1	2	0

Table 4.2.3. The difference between the solution quality [‰] between experiments with a 5-second time limit and the number of iterations without an improvement as the termination condition for particular metaheuristics starting from Johnson's schedule (J) or a list schedule (L) and the number of experiments (among 20) in which the run with a particular termination condition dominated the other one

Comparing the results obtained for different termination conditions, one could notice that although a 5-second time limit allowed the methods to achieve better results in most cases, this improvement was not significant (i.e. equal to fractions of per mill). Actually, for the majority of tests, the metaheuristics generated a schedule of the same quality independently of the termination condition applied (cf. Table 4.2.3, Figure 4.2.4 and Figure 4.2.5).

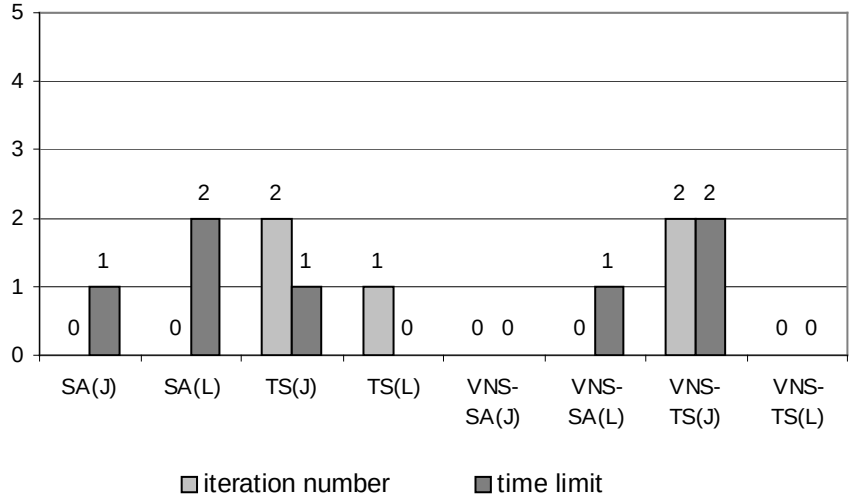

Figure 4.2.4. The number of experiments (among 20) with a better result obtained for particular termination conditions, i.e. the number of iterations without an improvement and a 5-second time limit, for particular metaheuristics starting from Johnson's schedule (J) or a list schedule (L)

For SA and VNS-SA, the time limit was always slightly more profitable than the number of iterations without an improvement limit, however such a difference between these termination

conditions was detected for only 1 or 2 instances depending on the initial solution. Since SA was a very fast algorithm, the large run time limit sometimes allowed it to leave a local optimum and to find a better final schedule, which would not be possible within a certain number of iterations without an improvement as the termination condition. On the contrary, the tabu search method and the VNS with TS, whose iterations were much more time consuming, behaved similarly for both termination conditions – at least no correlation could be observed.

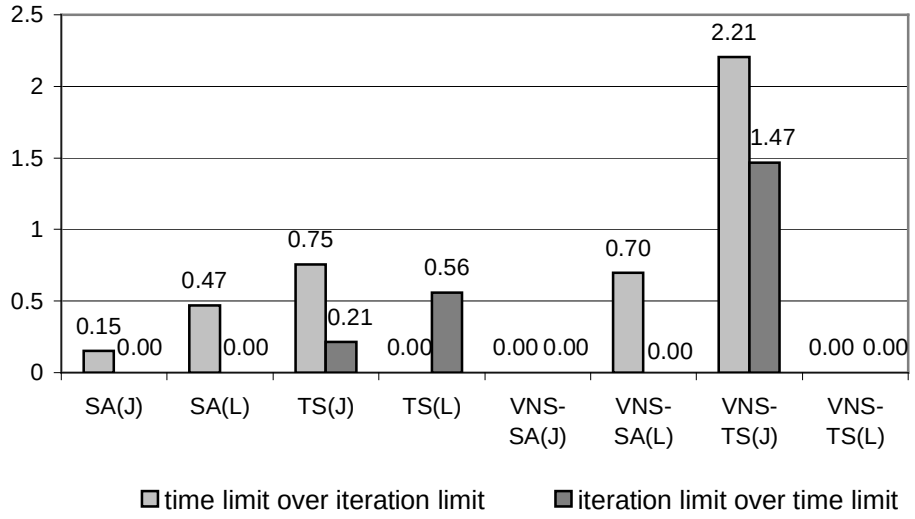

Figure 4.2.5. The average difference in [‰] between criterion values obtained for particular termination conditions, i.e. the number of iterations without an improvement and a 5-second time limit, for particular metaheuristics starting from Johnson's schedule (J) or a list schedule (L)

Summing up, the distance between the criterion values achieved for the two termination conditions investigated was very small. Taking into account only these computational experiments in which the results obtained for a 5-second time limit dominated the results achieved for the number of iterations without an improvement limit, the average difference in the criterion value over all metaheuristics was equal to 0.99‰. For the experiments in which the iteration limit was more profitable than the time limit, this average difference was a bit smaller and it equals 0.78‰. This fact can be explained by the tuning process performed for all metaheuristics investigated. It made it possible to determine a termination condition value (i.e. the number of iterations without an improvement) large enough to explore the solution space. Enforcing a longer run time could not considerably increase the efficiency of the methods (although it sometimes allowed the algorithms to leave a local optimum).

Obviously, the evaluation of the performance of the metaheuristics depends not only on the solution quality but also on its computational time requirements. The analysis of the time efficiency was restricted only to the experiments with the number of iterations without an improvement as the termination condition. Table 4.2.4 and Figures 4.2.6 and 4.2.7 present the results for particular numbers of jobs n as average values obtained for a few instances with the same value of n investigated twice with two different initial solutions (Johnson's and the list one).

n	average run time [sec.]				standard deviation			
	SA	TS	VNS-SA	VNS-TS	SA	TS	VNS-SA	VNS-TS
20	0.003	0.003	0.006	0.011	0.001	0.002	0.002	0.002
50	0.011	0.041	0.036	0.205	0.003	0.023	0.008	0.078
80	0.019	0.089	0.062	0.572	0.002	0.007	0.008	0.103
110	0.057	0.557	0.147	3.178	0.018	0.339	0.030	2.217
140	0.034	0.222	0.110	1.176	0.003	0.005	0.014	0.152

Table 4.2.4. The average run time in [sec.] for particular numbers of jobs n with the standard deviation value

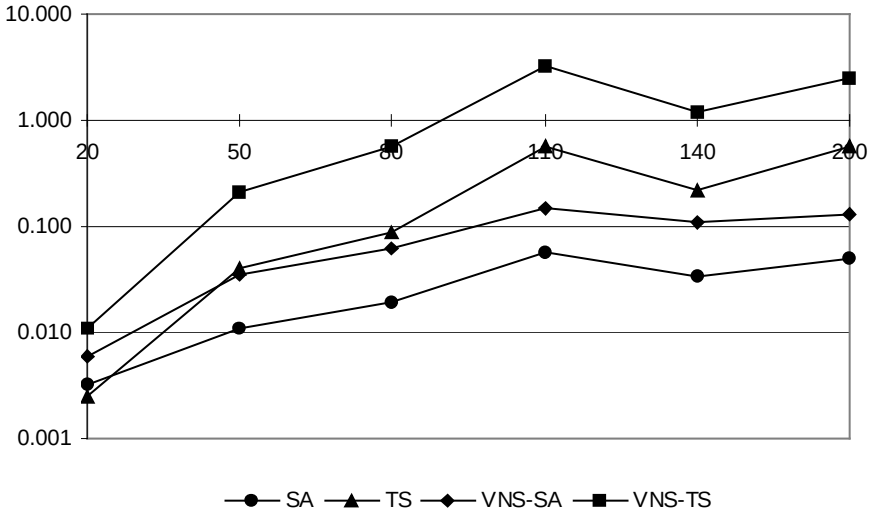

Figure 4.2.6. The average run time in [sec.] for particular numbers of jobs n with the logarithmic time axis

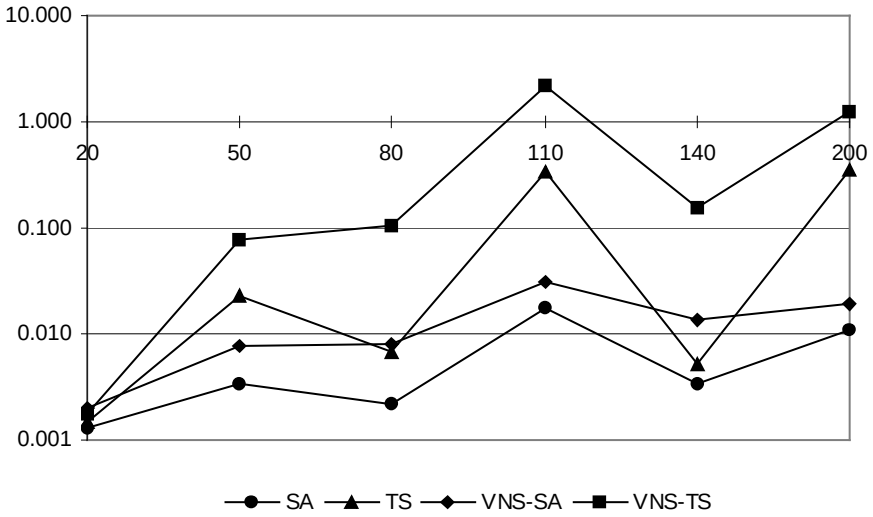

Figure 4.2.7. The standard deviation for the run time for particular numbers of jobs n with logarithmic Y-axis

Analyzing the test results, one could notice that the run time of all metaheuristics increased with the number of jobs (cf. Figure 4.2.6) which was caused by the termination condition applied for SA and TS (also for SA and TS embedded in VNS). Both algorithms terminated after reaching the given number of iterations without an improvement, which was settled after the tuning process as the number of jobs multiplied by some number (cf. Sections 4.1.1 and 4.1.2). Moreover, the computational experiments showed that the simulated annealing method was much more time

efficient than the tabu search algorithm. A single modification of a current solution, applied within SA, was less time consuming than generating the whole neighborhood in TS (cf. Table 4.2.4, Figure 4.2.6). Moreover, the SA behavior appeared to be more stable and more independent of the problem data which was reflected in a smaller standard deviation values than those achieved for TS (cf. Table 4.2.4, Figure 4.2.7).

Similar observations could be formulated for the variable neighborhood search method with two different local search procedures SA and TS, i.e. for VNS-SA compared to VNS-TS. Obviously the variable neighborhood search algorithm required more computational time than the simulated annealing or tabu search ones applied as stand-alone methods, because it repeated SA or TS a few times in order to restart the search from different initial solutions. According to the termination condition imposed (cf. Section 4.1.3), VNS stopped after performing 3 iterations without an improvement. This means that SA or TS were applied within VNS at least 3 times. Actually, on average, VNS-SA worked 3.08 times longer than SA, while VNS-TS run 5.79 times longer than TS. Because VNS-SA was usually not able to improve the solution found after the first SA run, it terminated after 3 iterations in total. In the case of VNS-TS, the schedule improvement happened more often than for VNS-SA and the total number of iterations was larger than the minimum possible one. Although VNS with TS performed more iterations during its search, this additional computational effort did not result in a significant solution quality improvement (cf. Table 4.2.1).

Comparing the average run time for the whole test set containing 20 instances of a different size (cf. Table 4.2.5, Figures 4.2.8, 4.2.9, 4.2.10), the superiority of the simulated annealing became even more visible. The tabu search overtook the simulated annealing for only one (cf. Figure 4.2.9) small problem instance (with 20 jobs), for which the computational time was almost immeasurable (cf. Figure 4.2.10). For the remaining 19 instances, SA found better schedules requiring about 7 times shorter time than TS.

		SA(J)	SA(L)	TS(J)	TS(L)	VNS-SA(J)	VNS-SA(L)	VNS-TS(J)	VNS-TS(L)
Run time for 20 test in [sec]	minimum	0.004	0.001	0.001	0.001	0.004	0.008	0.012	0.008
	average	0.031	0.028	0.232	0.246	0.088	0.082	1.423	1.124
	maximum	0.074	0.086	0.934	0.984	0.172	0.199	7.180	5.566
	stand. dev.	0.020	0.021	0.270	0.310	0.050	0.051	1.786	1.326
Run time divided by the shortest run time	minimum	1.11	1.15	3.38	2.88	2.16	2.48	12.00	8.00
	average	1.85	1.88	7.21	7.18	3.65	3.66	40.04	32.40
	maximum	4.00	4.00	26.69	26.00	6.38	8.00	118.31	88.83
	stand. dev.	1.04	1.22	5.10	5.37	0.84	1.52	27.03	18.97
	No of tests among 20 with the shortest run time	9	16	1	1	0	0	0	0

Table 4.2.5. The average run time for the 20-instance test set in [sec.] and the average value of the run time divided by the shortest run time for a particular instance starting from Johnson's schedule (J) or a list schedule (L) and the number of iterations without an improvement as the termination condition

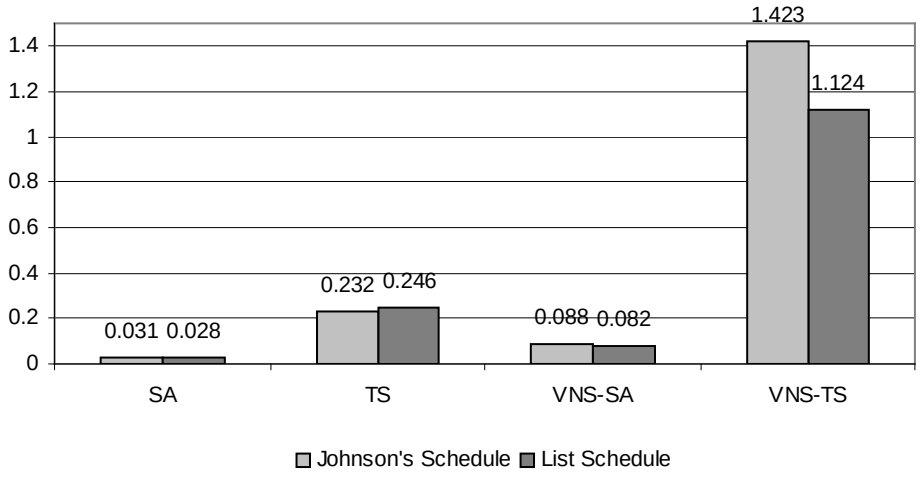

Figure 4.2.8. The average run time for 20-instance test set in [sec.] for the two different initial solutions and the number of iterations without an improvement as the termination condition

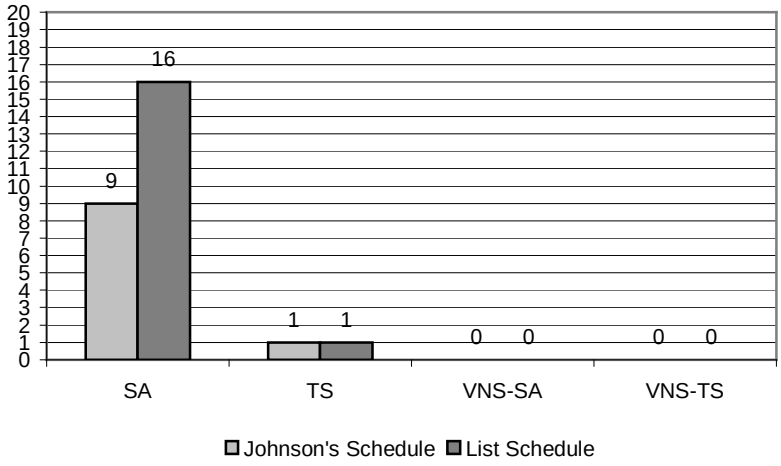

Figure 4.2.9. The number of test instances in which a particular method constructed a solution in shortest time (for 20 experiments) for two different initial solutions and the number of iterations without an improvement as the termination condition

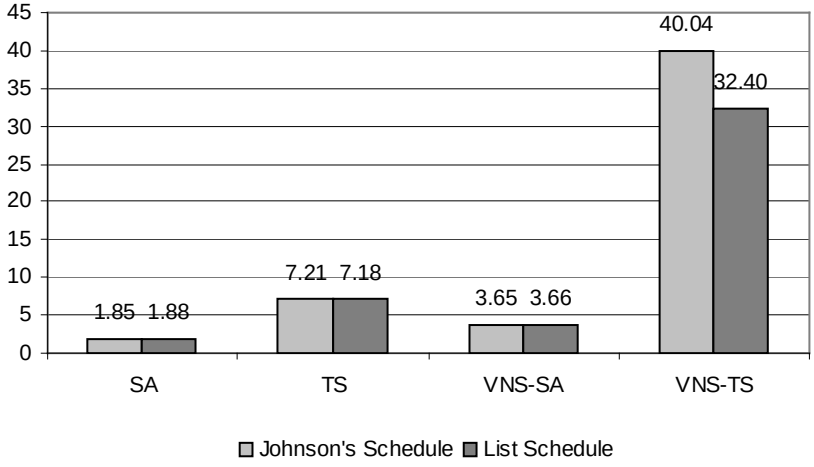

Figure 4.2.10. The average run time difference for 20-instance test set calculated as the quotient of the method run time and the shortest run time for particular instances for the two different initial solutions and the number of iterations without an improvement as the termination condition

The repeated application of SA within VNS consumed about 3 times more run time, while VNS-TS worked 30-40 times longer than SA (cf. Figure 4.2.10). As we have already mentioned, VNS-SA usually performed the minimal possible number of iterations, i.e. 3 iterations if no improvement in the SA solution was achieved. This means that VNS-SA run SA as a local search procedure 3 times on average. In the consequence, VNS-SA consumed a bit more than the tripled run time of SA applied as a stand-alone approach. On the contrary VNS-TS sometimes improved the solution generated by TS and the total number of iterations was larger than the allowed number of iterations without an improvement (i.e. three iterations). Moreover, the tabu search approach was much more time consuming (more than 7 times) and less stable than SA (cf. Figure 4.2.7) and the computational time for particular TS runs within VNS-TS could vary significantly.

Finally, based on the above analysis of the time requirements for particular metaheuristics (Figure 4.2.8 and Figure 4.2.10), it was possible to observe a slight influence of the initial schedule on the duration of the search process. Starting the exploration of the solution space from the list schedule, the metaheuristics terminated faster than starting their search from Johnson's schedule. The initial schedule of better quality (the list schedule) made the solution improvement more difficult and caused that the given limit of iterations without an improvement was exceeded quite early.

4.3. Comparison of Heuristic Methods to an Exact Approach

The second stage of the computational experiments was devoted to the comparison of the heuristic approaches to an exact enumerative method. Earlier research results obtained for problem $F2|d_j=d|Y_w$ [11] showed that the enumerative method (EM) was more time efficient than a pseudo-polynomial time dynamic programming one. EM investigates in $O(n2^n)$ time all possible solutions for the problem under consideration. It checks all possible subsets of early jobs (E) and executes them in Johnson's order. Then, EM considers each job among the remaining ones as the first late job J_x and completes a partial schedule with the remaining jobs from $J\setminus(E\cup\{J_x\})$ sequenced according to non-increasing weights. The outline of the presented enumerative method is given below.

for each set $E\subseteq J$ such that a partial schedule obtained by sequencing jobs from E in Johnson's order does not exceed d and Johnson's schedule for $E\cup\{J_x\}$ where $J_x \in J\setminus E$ exceeds d do
    for each job $J_x \in J\setminus E$ do
        construct a schedule by executing jobs from E in Johnson's order, followed by J_x and jobs from $J\setminus(E\cup\{J_x\})$ sequenced according to non-increasing weights;
    store the best solution constructed for set E, if it is better than the already found one.

The exact approach was tested against four metaheuristics: SA, TS, VNS-SA and VNS-TS as well as two other simple heuristics: the list scheduling algorithm with the maximum weight priority dispatching rule and Johnson's procedure used as an approximate method for $F2|d_j=d|Y_w$. Taking into account the exponential time complexity of the enumerative method, 16 small problem instances were used with:

- the number of jobs equal from 5 to 20 (with a unit increment),
- task processing times generated from the range [2, 20],
- job weights from the range [1, 5],

and the due date equal to 40% of the half of the total processing time of the jobs.

The average quality of particular heuristic solutions is presented in Table 4.3.1. Johnson's algorithm (JA) appeared to be the weakest method generating schedules of nearly 11% worse than the optimum. On the other hand, taking into account its neglectedly short run time, the simplicity of this procedure and the fact that it was designed for a different scheduling problem ($F2||C_{max}$), the results of JA might be quite satisfying for certain applications.

The list scheduling algorithm was able to construct an optimal solution for only 6 instances among 16 test problems. Moreover, it found an optimum mostly for small instances with 6,7,8 and 9 jobs. The simulated annealing and tabu search methods reached the optimal criterion value for 10 instances. Applying these procedures within the variable neighborhood search framework increased the number of optimums found to 11 and 12, respectively. In general, the best metaheuristic solution hds the same criterion value as an optimal solution constructed by the enumerative algorithm for 13 tests among 16 (cf. Figure 4.3.1).

		JA	LA	SA	TS	VNS-SA	VNS-TS	BMH
all tests	minimum	1.26	0.00	0.00	0.00	0.00	0.00	0.00
	average	10.70	0.81	0.40	0.40	0.27	0.25	0.04
	maximum	21.15	2.46	2.26	2.46	2.25	2.46	0.20
	standard dev.	6.45	1.21	0.61	1.25	0.56	0.62	0.07
	No of optimal solutions in 16 tests	0	6	10	10	11	12	13
non-optimal results	minimum	1.26	0.59	0.39	0.18	0.20	0.18	0.18
	average	10.70	1.35	1.00	1.21	0.81	0.95	0.19
	maximum	21.15	2.46	2.26	2.46	2.25	2.46	0.20
	standard dev.	6.45	1.14	0.61	1.50	0.74	0.93	0.01

Table 4.3.1. The distance to the optimal criterion value for particular methods and for the best metaheuristic in a certain test (BMH) in [%] for all tests and for these tests for which a particular method did not find the optimum

The specificity of the problem under consideration, $F2|d_j=d|Y_w$, made the search process difficult. Taking into account the fact that all early jobs have to be scheduled in Johnson's order, the crucial decision is to select activities performed before the common due date d. At this stage of the computational experiments, the initial solution for all metaheuristic methods was generated by the list algorithm, which selected early jobs according to their weights. When particular jobs differ only slightly in their processing times and weights, a further improvement of the criterion value by the metaheuristics is difficult, since an optimal solution is a specific sequence of jobs, which does not differ to much from the initial one. From this point of view, the efficiency of the metaheuristic methods proposed was quite satisfying.

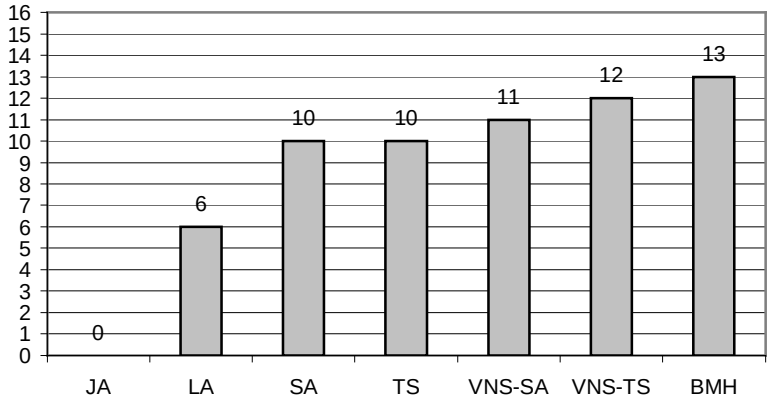

Figure 4.3.1. The number of optima found in 16 tests for particular methods and the best metaheuristic result (BMH)

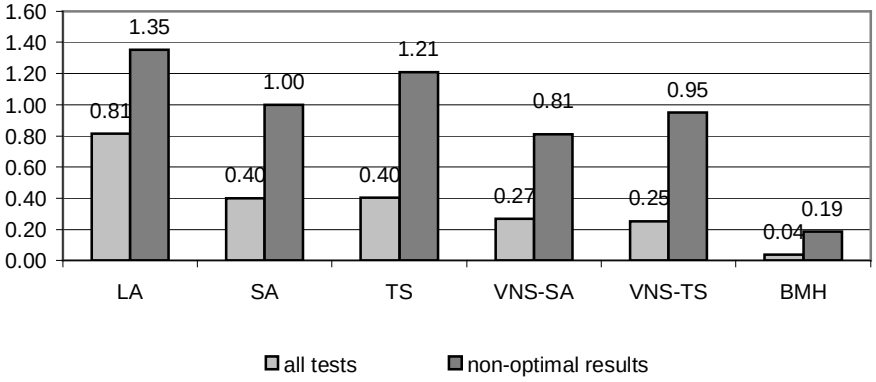

Figure 4.3.2. The distance to the optimal criterion value for particular methods and the best metaheuristic in a certain test (BMH) in [%] for all tests and for those tests for which a particular method did not find the optimum

Actually, these observations were confirmed by the analysis of the average distance to the optimal criterion value (cf. Figure 4.3.2) calculated for all 20 tests and, especially, for only these instances for which particular methods did not reach the optimum.

The best metaheuristic solution differed from the optimum for only 0.19%, while for the list scheduling one this value equaled to 1.35%.

Among metaheuristics, the highest performance in terms of the schedule quality was observed for VNS-SA, for which the distance to the optimum was equal to 0.81%. As we noticed in Section 4.2, the repeated use of SA within VNS made it possible to improve slightly the quality of schedules constructed by the simulated annealing approach applied as a stand-alone method. The SA distance to the optimum equaled to only 1%.

Similarly as in the computational experiments with large instances reported in Section 4.2, for small problem instances the tabu search algorithm appeared to be less efficient than SA. However, the difference was not so significant, about 0,21%.

VNS-TS behaved for small numbers of jobs much better than for the job sets of larger cardinality. The quality of VNS-TS was comparable to VNS-SA and SA (cf. Figure 4.3.2). When the number of jobs was small, the size of the neighborhood for a current solution was also small and TS, applied as a stand-alone algorithm or as a local search procedure within VNS, worked more efficiently.

number of jobs	EM [msec.]	SA [msec.]	TS [msec.]	VNS-SA [msec.]	VNS-TS [msec.]
5	0.015	0.001	*0.001*	*0.001*	*0.001*
6	*0.001*	*0.001*	*0.001*	0.001	0.001
7	*0.001*	0.001	*0.001*	0.001	0.001
8	*0.001*	*0.001*	0.001	0.001	0.001
9	*0.001*	0.001	0.001	0.001	*0.001*
10	*0.001*	0.001	0.001	0.002	0.001
11	0.015	0.001	*0.001*	0.002	0.002
12	0.031	0.001	0.001	0.001	0.003
13	0.187	0.001	0.001	0.002	0.004
14	0.218	0.001	0.001	0.001	0.004
15	0.328	*0.001*	0.001	0.003	0.011
16	0.89	0.001	0.003	0.004	0.012
17	1.359	0.001	0.002	0.003	0.007
18	8.843	0.001	0.003	0.004	0.01
19	52.375	0.002	0.001	0.006	0.01
20	71.281	0.001	0.004	0.003	0.021

Table 4.3.2. Run time for particular methods in [msec.]
(results in italic denote that the run time was immeasurably short)

The run time comparison (cf. Table 4.3.2 and Figure 4.3.3) confirmed the observations formulated above on the efficiency of particular metaheuristics, but first of all it underlined the necessity of applying this kind of search procedures for hard problems. The enumerative approach ensured the optimality of the solutions constructed, but it required a huge computational effort, increasing rapidly with the problem size. On the contrary, the metaheuristic run time was low and it increased very slowly with the number of jobs. In the computational experiments reported, the exponential explosion for the exact approach was observed already for 18 jobs.

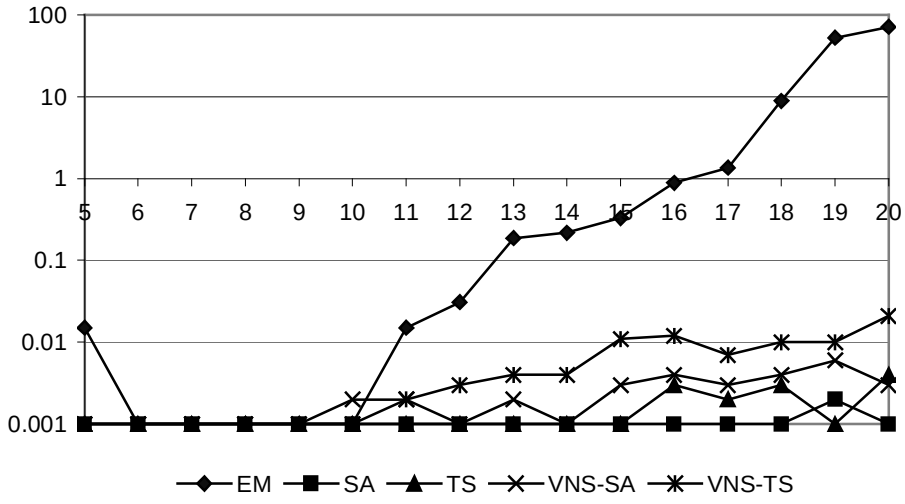

Figure 4.3.3. Run time in [msec.] for particular methods

5. Conclusions

The research presented completes the studies on the two-machine flow shop problem with a common due date and the weighted late work criterion, $F2|d_j=d|Y_w$. The theoretical investigation resulted in the NP-completeness proof for its decision counterpart and the proposal of a pseudo-polynomial time dynamic programming approach, that allowed us to classify the case as binary NP-hard [10]. The first stage of computational experiments made it possible to verify the applicability of the DP procedure in practice and showed a high efficiency of the list scheduling algorithm [11]. At the second stage of computational studies reported in this paper, more advance solution techniques were proposed: simulated annealing, tabu search and variable neighborhood search methods, based on the specific structure of an optimal solution for the problem under consideration. These trajectory metaheuristics were compared in the computational experiments one to each other as well as to the list scheduling approach, Johnson's method applied as a heuristic for $F2|d_j=d|Y_w$ and to an exact enumerative method.

The main experiments were preceded by the extensive tuning process in order to determine the best control parameter settings for particular metaheuristic methods. The difference in the performance for a single problem instance and different control parameter settings reached almost 10% for the simulated annealing and 7% for the tabu search approach whic confirms the importance of the tuning process. Moreover, these preliminary experimental results made it possible to formulate interesting observations on the SA, TS and VNS-SA as well as VNS-TS behavior.

Then, we compared the metaheuristic approaches to the list scheduling algorithm for large instances in terms of the number of jobs. Despite the high quality of the list solution, the metaheuristic methods were still able to improve its quality. Furthermore, the test results showed that the simulated annealing significantly dominates the tabu search strategy for the scheduling case under consideration. SA generated better schedules in shorter time than TS. The fast SA method moving from one solution to another as a result of a single solution modification, was more efficient than TS generating the whole neighborhood in order to select the next schedule for the analysis. Moreover, restarting SA within the variable neighborhood framework made an additional solution improvement possible. A similar effect was observed for TS and VNS-TS, although it was less visible. In the computational experiments with small instances in terms of the number of jobs, the differences among particular metaheuristics became less apparent, since all trajectory methods proposed generated optimal solutions for most test sets. Their time requirements were incomparably small with regard to the enumerative algorithm of exponential time complexity.

The experience gained during the research on problem $F2|d_j=d|Y_w$ provided many useful hints for future work on other scheduling problems with the late work performance measure.

Acknowledgement. We would like to thank to Michal Glowinski for his effort in implementing and testing the approaches investigated within the presented research. The fourth author has been supported by INTAS (project 03-51-5501).

References

[1] R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende, W.R. Stewart JR.: Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics* 1 (1995) 9-32

[2] Ch. Blum, A. Roli: Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys* 35/3 (2003) 268-308

[3] J. Błażewicz: Scheduling preemptible tasks on parallel processors with information loss. *Technique et Science Informatiques* 3/6 (1984) 415-420

[4] J. Błażewicz, K. Ecker, E. Pesch, G. Schmidt, J. Węglarz: *Scheduling Computer and Manufacturing Processes*. 2nd edn. Springer, Berlin-Heidelberg-New York (2001)

[5] J. Błażewicz, G. Finke: Minimizing mean weighted execution time loss on identical and uniform processors. *Information Processing Letters* 24 (1987) 259-263

[6] J. Błażewicz, E. Pesch, M. Sterna, F. Werner: Total late work criteria for shop scheduling problems. In: K. Inderfurth, G. Schwödiauer, W. Domschke, F. Juhnke, P. Kleinschmidt, G. Wäscher (Eds.): *Operations Research Proceedings* 1999. Springer, Berlin (2000) 354-359

[7] J. Błażewicz, E. Pesch, M. Sterna, F. Werner: Revenue management in a job-shop: a dynamic programming approach. *Preprint* Nr. 40/03, Otto-von-Guericke-University Magdeburg (2003)

[8] J. Błażewicz,, E. Pesch, M. Sterna, F. Werner: Open shop scheduling problems with late work criteria. *Discrete Applied Mathematics* 134 (2004) 1-24

[9] J. Błażewicz,, E. Pesch, M. Sterna, F. Werner: Flow shop scheduling with late work criterion – choosing the best solution strategy. *Lecture Notes in Computer Science* 3285 (2004) 68-75

[10] J. Błażewicz,, E. Pesch, M. Sterna, F. Werner: The two-machine flow-shop problem with weighted late work criterion and common due date. *European Journal of Operational Research* 165/2 (2005) 408-415

[11] J. Błażewicz,, E. Pesch, M. Sterna, F. Werner: A comparison of solution procedures for two-machine flow shop scheduling with late work criterion. *Computers and Industrial Engineering* (to appear)

[12] P. Brucker: *Scheduling Algorithms*. 2nd edn. Springer, Berlin-Heidelberg-New York (1998)

[13] B. Chen, C.N. Potts, G.J. Woeginger: A review of machine scheduling. In: D.-Z. Du, P.M. Pardalos (Eds.): *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, Boston (1998)

[14] Y. Crama, A. Kolen, E. Pesch: Local Search in Combinatorial Optimization. *Lecture Notes in Computer Science* 931 (1995) 157-174

[15] M.R. Garey, D.S. Johnson: *Computers and Intractability*. W.H. Freeman and Co., San Francisco (1979)

[16] F. Glover, M. Laguna: *Tabu Search*. Kluwer Academic Publishers, Boston (1997)

[17] P. Hansen, N. Mladenović: An introduction to variable neighbour search. In: S. Voss, S. Martello, I. Osman, C. Roucairol (Eds.): *Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Boston (1999) Chapter 30, 433-458

[18] A.M.A. Hariri, C.N. Potts, L.N. Van Wassenhove: Single machine scheduling to minimize total late work. *INFORMS Journal on Computing* 7 (1995) 232-242

[19] R. Haupt: A survey of priority rule – based scheduling. *OR Sektrum* 11 (1989) 3-16

[20] J.N. Hooker: Testing heuristics: we have it all wrong. *Journal of Heuristics* 1 (1995) 33-42

[21] D.S. Hochbaum, R. Shamir: Minimizing the number of tardy job unit under release time constraints. *Discrete Applied Mathematics* 28 (1990) 45-57

[22] R.B. Kethley, B. Alidaee: Single machine scheduling to minimize total late work: a comparison of scheduling rules and search algorithms. *Computers and Industrial Engineering* 43 (2002) 509-528

[23] S.M. Johnson: Optimal two- and three-stage production schedules. *Naval Research Logistics Quarterly* 1 (1954) 61-68

[24] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi: Optimization by simulated annealing. *Science* 220/4598 (1983) 671-680

[25] M.Y. Kovalyov, C.N. Potts. L.N. Van Wasenhove: A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work. *Mathematics of Operations Research* 19/1 (1994) 86-93

[26] J.Y-T. Leung, V.K.M. Yu, W-D. Wei: Minimizing the weighted number of tardy task units. *Discrete Applied Mathematics* 51 (1994) 307-316

[27] J.Y-T. Leung: Minimizing total weighted error for imprecise computation tasks and related problems. In: J.Y.T. Leung (Ed.), *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Boca Raton (2004) Chapter 34, 1-16

[28] M. Pinedo, X. Chao: *Operation Scheduling with Applications in Manufacturing and Services*. Irwin/McGraw-Hill, Boston (1999)

[29] C.N. Potts, L.N. Van Wassenhove: Single machine scheduling to minimize total late work. *Operations Research* 40/3 (1991) 586-595

[30] C.N. Potts, L.N. Van Wassenhove: Approximation algorithms for scheduling a single machine to minimize total late work. *Operations Research Letters* 11 (1991) 261-266

[31] M. Sterna: *Problems and Algorithms in Non-Classical Shop Scheduling*. Scientific Publishers of the Polish Academy of Sciences, Poznań (2000)