Control and Cybernetics

vol. **40** (2011) No. 4

Linear programming & metaheuristic approach for scheduling in the hybrid flowshop with resource constraints*

by

Ewa Figielska

Warsaw School of Information Technology
Newelska 6, 01- 447 Warsaw, Poland
e-mail: figielsk@wit.edu.pl

Abstract: This paper deals with the problem of preemptive scheduling in a two-stage flowshop with parallel unrelated machines and additional renewable resources. The objective is the minimization of makespan. The problem is NP-hard. Heuristic algorithms are proposed which join the linear programming based procedures with metaheuristic algorithms: genetic, simulated annealing and tabu search algorithm. The performance of the proposed algorithms is experimentally evaluated by comparing the solutions with a lower bound on the optimal makespan. Results of a computational experiment show that these algorithms are able to produce good solutions in short computation time and that the metaheuristics significantly improve the results for the most difficult problems.

Keywords: flowshop, parallel machines, resource constraints, heuristics, linear programming, genetic algorithms, simulated annealing, tabu search.

1. Introduction

This paper deals with the problem of preemptive scheduling in the two-stage flowshop with unrelated parallel machines and additional renewable resource constraints in the first stage and a single machine in the second stage. A number of jobs have to be processed in two stages, each job being processed first in stage 1 and then in stage 2. A job upon finishing its processing in stage 1 is ready to be processed in stage 2; it may be processed in stage 2 when the machine is available there, or it may reside in a buffer space of unlimited capacity following stage 1 until the machine in stage 2 becomes available. Jobs during their processing in the first stage, besides the machines, use additional renewable resources which are available in limited quantities at any time. A job does not need a resource

*Submitted: June 2010; Accepted: September 2011

or requires one unit of this resource (0-1 resource requirements). Processing a job on a machine may be interrupted at any moment and resumed later on the same or another machine. We assume that there is no setup time between the operations. This is motivated by the fact that in automated manufacturing systems, automatic devices quickly interchange tools, so the time needed to set up a machine to execute a job is negligible (see, e.g., Crama, 1997). A job (part of a job) can be processed on any machine, its processing rates may be different on different machines and there is no relationship between them. The aim is to find a feasible schedule in the flowshop, minimizing the makespan, which is defined as the maximum job completion time in stage 2.

The problem considered is NP-hard in the strong sense, since the problem of preemptive scheduling in the two-stage flowshop with two identical parallel machines in one stage and one machine in the other one is NP-hard in the strong sense (Hoogeveen et al., 1996).

For solving this problem, we combine the linear programming based method proposed in Figielska (2008) with metaheuristics. In Figielska (2008), for solving this problem, the two-phase method proposed in Słowiński (1981) for the one stage system has been adapted for the minimization of the makespan in the two-stage flowshop, and five heuristic algorithms have been developed. For improving the results provided by these algorithms we use three metaheuristics: a genetic algorithm (GA), a simulated annealing (SA) algorithm and a tabu search (TS) algorithm. So, in this paper, 15 algorithms are designed. The performance of these algorithms, measured by the percentage deviation of the heuristic makespan from the lower bound on the optimal makespan is evaluated and compared to the performance of the algorithms developed in Figielska (2008).

In the one stage environment, the problem of preemptive scheduling for parallel unrelated machines under resource constraints in the case of 0-1 resource requirements has been solved in Słowiński (1980, 1981), where a two-phase approach was proposed, in which the solution of an LP problem provides input data for a procedure finding an optimal schedule. Some extensions of this approach were proposed in de Werra (1984, 1988). The survey of scheduling under resource constraints can be found in Błażewicz et al. (1987). Many papers address flowshop with parallel machines, but they consider mainly the case with identical machines (e.g. Gupta, 1988; Chen, 1995; Haouari and M'Hallah, 1997; Brah and Loo, 1999; Jin et al., 2006). A handful of papers address scheduling in the flowshop with unrelated machines (e.g. Adler et al., 1993; Suresh, 1997; Low, 2005). The literature reviews on scheduling in a flowshop with parallel machines can be found in Lin and Zhang (1999), Vignier et al. (1999), Cheng et al. (2000), Gupta and Stafford (2006).

Metaheuristic algorithms, such as tabu search (Glover, 1989), simulated annealing (Metropolis, 1953), and genetic algorithms (Holland, 1975), have been applied to solving a variety of scheduling problems and they were proved to be able to provide near optimal solutions in reasonable time. In the area of

flowshop with parallel machines, for example, simulated annealing and genetic algorithms were used for problems with setup times (Aldowaisan and Allahverdi, 2003; Low, 2005), a tabu search algorithm was applied to a problem with limited buffer capacities (Wardono and Fathi, 2004), a genetic algorithm was developed for scheduling multiprocessor tasks (Oguz and Erkan, 2005), and simulated annealing and genetic algorithms were used for a problem with resource constraints in the case of arbitrary resource requirements (Figielska, 2009).

The problem of scheduling in a multiprocessor flowshop arises in real-life systems that are encountered in a variety of industries, e.g. in chemical, polymer, petrochemical industries (Salvador, 1973). The multiprocessor scheduling problem is also met in computer systems and telecommunication networks (Brah, 1988). In the multiprocessor flowshop there are stages with parallel machines. At each stage with parallel machines jobs can be processed through any one of these machines. Because jobs that are simultaneously processed on parallel machines may use the same resource, the problem of resource constrained scheduling arises when the amount of the available resource is limited. This takes place when, for example, the number of workers attending the machines, or the number of tools that are used by simultaneously executed jobs, is limited. Resource requirements of 0-1 type can be met, for instance, in computer systems in which one peripheral device (e.g. a printer) is additionally required to perform a job (Kellerer and Strusevich, 2003). The problems with preemptive jobs are common in mass production of large numbers of products which can be processed in parts or when an article is produced in a great amount, for example in the textile industry (Serafini, 1996), where the processing of any job (the article to be woven) on one of the parallel machines (the looms) may be interrupted (preempted) and resumed on the same or the other machine. Preemptive scheduling is useful in the automated manufacturing systems where the loss of the time associated with the preemptions of jobs is small compared with the gain in the schedule makespan.

2. Problem description

There are n preemptive jobs to be processed in two stages in the same technological order, first in stage 1 then in stage 2. In stage 1 there are m parallel unrelated machines, stage 2 has one machine. The buffer space between the stages has unlimited capacity. Jobs for their processing in stage 1, besides machines, require additional resources. There are l types of renewable resources. A resource of type r ($r = 1, \ldots, l$) is available in the amount limited to W_r units at a time. The total usage of resource r at any moment by jobs simultaneously executed on parallel machines cannot exceed the availability of this resource. Each job, during its processing in stage 1, uses 0 or 1 unit of the resource of each type. The processing time of job j ($j = 1, \ldots, n$) is equal to p_{ij} if it is executed on machine i ($i = 1, \ldots, m$) in stage 1, and the processing time of job j in stage 2 is equal to s_j. The objective is to find a feasible schedule, which

minimizes makespan, which is equal to the maximum job completion time at stage 2, $C_{\max}$.

3. Heuristic algorithms

The proposed heuristic first solves to optimality the resource constrained unrelated machine scheduling problem for stage 1, then, given a schedule for stage 1, the schedule for stage 2 is constructed. The process of solving the problem at stage 1 proceeds in two steps. In step 1, the two-phase linear programming based procedure, which uses some selection rules for jobs simultaneously executed in successive periods of time (Linear Programming & Selection Rules (LPSR) procedure), proposed in Figielska (2008) creates a schedule in the form of a sequence of partial schedules. In a partial schedule at most m (m is the number of machines) jobs are assigned to m machines for simultaneous processing during some period of time so that resource constraints are satisfied at every moment. The sequence of the partial schedules (each of which is given a unique index) determines the order in which they are executed. Each consecutive partial schedule contains jobs, which are selected so as to obtain as small as possible makespan in the flowshop. In step 2, one of the metaheuristics: genetic algorithm, simulated annealing or tabu search, changes the ordering of the partial schedules obtained in step 1, so as to minimize the makespan in the flowshop. In all the metaheuristics the solution is represented by a string of integers, which are the indices of consecutive partial schedules.

This heuristic can be outlined as follows:

I Find the first stage schedule.

Step 1. Find a schedule by a linear programming based algorithm.

a) A linear programming (LP) problem with relaxed resource constrains is solved to minimize time T^* needed for finishing all jobs at stage 1 of the flowshop (see Słowiński, 1980, 1981). As a result, the values of times t^*_{ij} ($i = 1, \ldots, m$, $j = 1, \ldots, n$) (t^*_{ij} is the time during which job j is processed on machine i) are obtained.

b) Using the values of t^*_{ij} as well as the values of s_j (s_j is the processing time of job j at stage 2), weights, w_j, for all jobs are determined on the basis of five different rules: (1) $w_j = 1$, (2) $w_j = \min_{k=1,\ldots,n}(Z_k)/Z_j$, (3) $w_j = s_j/\max_{k=1,\ldots,n}(s_k)$, (4) $w_j = (s_j/Z_j)(\min_{k=1,\ldots,n}(Z_k)/\max_{k=1,\ldots,n}(s_k))$, (5) $w_j = \min_{k=1,\ldots,n}(Z_k)/Z_j + 1$ if $Z_j \leq s_j$ and $s_j/\max_{k=1,\ldots,n}(s_k)$ if $Z_j > s_j$, where Z_j is the processing time of job j at stage 1, $Z_j = \sum_{i=1}^{m} t^*_{ij}$ (in (1), (2), (3) and (4) the maximal value of w_j is equal to 1, in (5) the maximal w_j is equal to 2 if $Z_j \leq s_j$, and 1 if $Z_j > s_j$).

c) A schedule of length equal to T^* time units is created in the form of a sequence of partial schedules using the values of t^*_{ij}

and w_j. The consecutive partial schedules are created in subsequent iterations of an iterative procedure. Assignment of jobs to machines in a partial schedule is found maximizing the weighted assignment $\sum_{i=1}^{m}\sum_{j=1}^{n} w_j v_{ij}^b$ ($v_{ij}^b = 1$ if job j is processed on machine i in partial schedule S^b, and 0 otherwise) under resource constraints. In each created partial schedule conditions on optimality formulated in Słowiński (1981) are satisfied. The details and related IP models can be found in Figielska (2008).

Step 2. Improve the schedule by a metaheuristic algorithm.
One of the metaheuristic algorithms: a genetic algorithm, a simulated annealing algorithm or a tabu search algorithm, introduces changes in the ordering of the partial schedules so as to decrease the makespan in the flowshop as much as possible.

II Find the second stage schedule
Given the schedule for stage 1, completion times at stage 1 are determined. The schedule for stage 2 is constructed taking into account the completion times of jobs in stage 1 (which are equal to ready times of jobs in stage 2) and processing times in stage 2. The makespan in the flowshop is equal to the maximum job completion time in stage 2.

In Figielska (2008) five heuristic algorithms A1-A5, which use 5 different job selection rules, have been developed.

In this paper, we propose three metaheuristics over each of the five different rules given in Figielska (2008) as indicated in Table 1.

Table 1. Heuristic algorithms

	rule 1	rule 2	rule 3	rule 4	rule 5
GA	A1G	A2G	A3G	A4G	A5G
SA	A1S	A2S	A3S	A4S	A5S
TS	A1T	A2T	A3T	A4T	A5T

In the remainder of this section we present the metaheuristic algorithms, which are used in Step 2 of the heuristic for finding the best ordering of the partial schedules.

3.1. Genetic algorithm

A GA starts with an initial population composed of a number of candidate solutions (called chromosomes). A chromosome is represented by a string of numbers called genes. Each chromosome in the population is evaluated according to some fitness measure. Certain pairs of chromosomes (parents) are selected on the basis of their fitness. Each of these pairs combines to produce

new chromosomes (offspring) and some of the offspring are randomly modified. A new population is then formed replacing some of the original population by an identical number of offspring. The process is repeated until a stopping criterion is met.

The applied GA can be outlined as follows.

Let $P(t)$ denote the population in iteration t, and *pop_ size* be the population size.

1. Generate and evaluate the initial population $P(t)$, $t = 0$.
2. Repeat the following steps until stopping condition is satisfied.
   - 2.1. Repeat the following loop *pop_ size*/2 times (*pop_ size* is an even number).
     - 2.1.1. Select two parents from $P(t)$.
     - 2.1.2. Apply the crossover operator over the parent chromosomes and produce 2 offspring chromosomes.
     - 2.1.3. Apply the mutation operator over the offspring.
     - 2.1.4. Copy the offspring to population $P(t + 1)$.
   - 2.2. Evaluate $P(t + 1)$.
   - 2.3. Replace the worst chromosome of $P(t + 1)$ by the best chromosome found so far.
   - 2.4. Set $t = t + 1$.
3. Return the best chromosome found.

The factors which characterize the GA applied to the problem considered in this paper are determined as follows.

Solution representation. A solution to the sequencing problem solved by the GA is coded as a single chromosome whose genes represent the indices of the partial schedules.

Initial population. An initial population of chromosomes is generated on the basis of the solution (a sequence of the partial schedules), which has been obtained in Step 1 of the heuristic by the LPSR procedure. In the initial population, there is one chromosome (Ch1) which represents the ordering of the partial schedules obtained by the LPSR procedure. The rest of the initial population comprises the neighbors of Ch1. Each neighbor of Ch1 is created by removing a randomly chosen gene from its position and putting it to another randomly chosen position.

Evaluation. To measure the fitness of a chromosome, the value of the objective function, which is equal to the makespan in the flowshop, is used. For each partial schedule sequence (chromosome) generated in the search process, a schedule for the flowshop is constructed and its makespan is calculated.

Parent selection. As a selection method the binary tournament selection (Goldberg, Korb and Deb, 1989) is used.

Crossover. The crossover operation uses the two-point crossover operator PMX (Goldberg, 1989) for each pair of parent chromosomes with a probability P_{crs} (crossover probability).

Mutation. In the mutation operation, genes of each chromosome in a population are considered one by one, and the gene being considered is removed from its position and put into another randomly chosen position of the same chromosome with a probability P_{mut} (mutation probability).

Stopping condition. The search process terminates after visiting the required number of solutions.

In our implementation, the population size is set at 20, $P_{crs} = 0.8$ and $P_{mut} = 0.05$. The stopping criterion is set at 3000 visited solutions.

3.2. Simulated annealing

An SA procedure starts from an initial solution, and then, in each successive iteration, a new solution S' in the neighborhood of the current solution S is generated. Next, the values of energy function $E(S)$ and $E(S')$ are determined (usually they are equal to objective function values) and the change in the energy function value, $\Delta = e(S') - E(S)$, is calculated. For a minimization problem the move to a new solution is accepted with probability $P = \min\{1, \exp(-\Delta/T)\}$, where T is a control parameter called temperature. $P = 1$ (the move is accepted) if $\Delta \leq 0$ ($E(S') \leq E(S)$). If $\Delta > 0$, $\exp(-\Delta/T)$ is a decreasing function of Δ, so slightly worse solutions have a higher probability of being accepted than much worse solutions. This possibility of accepting worse neighbor solution allows for avoiding being trapped in local minima. On the other hand, the search process should converge to a good local minimum. This is achieved by gradually lowering the temperature as the search makes progress. At each temperature, a search is carried out for a certain number of iterations, called the Markov chain length.

While designing an SA algorithm, some decisions have to be made, namely, decisions specific to the problem at hand and decisions internal to the annealing process.

The problem specific decisions are as follows.

Solution representation. A solution to the sequencing problem which is dealt with is coded as a sequence of the indices of partial schedules.

Initial solution generation. An initial solution is the solution obtained in Step 1 by the LPRS procedure.

Neighbor generation. A neighbor of the current solution is obtained by removing a randomly chosen element (a partial schedule index) from one position and putting it into another randomly chosen position.

Energy function calculation. The energy function value is equal to the flowshop makespan.

The decisions internal to the annealing process (which are usually called cooling strategy) are presented below (Aarts and Laarhoven, 1987).

Initial temperature. The initial value of the temperature is calculated from the equation: $T_0 = \bar{\Delta}^+ / ln(\eta^{-1})$, where $\bar{\Delta}^+$ is the average increase in the energy function value for a number of random moves, and η is the initial acceptance ratio defined as the number of accepted moves divided by the number of proposed moves. In our implementation we demand that at least 90% moves be accepted at the start of the algorithm, so we set $\eta = 0.9$. $\bar{\Delta}^+$ is calculated for 30 randomly generated neighbors of the initial solution.

Decrement of the temperature. Temperature T is progressively reduced (starting from the initial value T_0) using a reduction factor κ ($0 < \kappa < 1$). In our implementation κ is set at 0.98.

The length of Markov chains. The length of Markov chains (the number of moves generated for a fixed value of the temperature) is determined so that the number of accepted moves for each temperature is equal to a fixed number μ. However, since moves are accepted with decreasing probability, to avoid too long Markov chains for small values of temperature at the end of the search, the number of moves for each T is limited to some constant $\bar{M}$. We assume $\mu = 10$ and $\bar{M} = 30$.

Stopping condition. The search process terminates after visiting the required number of solutions which is set at 3000.

The SA algorithm can be outlined as follows.

1. Get the initial temperature T_0, and set $T = T_0$. Get the values of κ, μ and $\bar{M}$.
2. Get the initial solution S .
3. Repeat until the stopping criterion is satisfied:
   - 3.1. Set $ac\_counter = 0$ and $counter = 0$.
   - 3.2. Repeat until ($ac\_counter \geq \mu$) or ($counter \geq \bar{M}$):
     - 3.2.1. Generate a neighbor S' of S.
     - 3.2.2. Calculate $\Delta = E(S') - E(S)$.
     - 3.2.3. If $\Delta \leq 0$, set $S = S'$, $E(S) = E(S')$ and $ac\_counter = ac\_counter+1$. Update the best solution (if necessary).
     - 3.2.4. Else if $\Delta > 0$ and random$[0, 1) < \exp(-\Delta/T)$, set $S = S'$, $E(S) = E(S')$ and $ac\_counter = ac\_counter+1$
     - 3.2.5. Set $counter = counter + 1$.
   - 3.3. Reduce the temperature by setting $T = \kappa T$.
4. Return the best solution found.

3.3. Tabu search

A TS algorithm is an iterative procedure. It starts from an initial solution (which is considered as a basic solution), generates a basic solution neighborhood which is a set of solutions created by the moves introducing small changes to the basic solution, and searches through this neighborhood for the best solution. The search is then repeated in the neighborhood of the best solution found previously (which now becomes a basic solution), and the process is continued. As the best solution in the neighborhood of a basic solution may be worse than the basic solution, the solution has a chance to escape from a local optimum. However, such escaping from a local optimum may be difficult because of the fact that cycling may occur, i.e. the situation where the algorithm returns to the same solution every few iterations. To avoid cycling some moves are forbidden (tabu) for a certain number of iterations. They are kept on a tabu list. The tabu list contains usually a small number of most recent tabu moves.

Below, we present the factors which characterize the TS algorithm applied to the considered problem.

Solution representation, initial solution generation and objective function calculation are the same as in the SA algorithm.

Neighborhood and moves. A neighborhood is a subset $Q'(S)$ of the set $Q(S)$ of all the neighbors of a basic solution S. A neighbor of a basic solution is obtained by removing a randomly chosen element (a partial schedule index k) from its position (denoted by x) and inserting it in another randomly chosen position (denoted by y) (the content of each position between x and y (including y) is shifted one position to the left (if $x < y$) or to the right (if $x > y$)). A move leading from a solution to a neighboring one is defined as a 3-tuple: the index of a partial schedule, k, a position from which it is removed, x, and a position in which it is inserted, y.

Tabu list. A new item (a move) is added to a tabu list each time a new basic solution is found. If (k, x, y) is the move leading from the previous basic solution to the new one, a reversed move (k, y, x) becomes forbidden and is added to the tabu list (the oldest item is removed from the list, if the maximum list size is exceeded).

Stopping condition. The search process terminates after visiting the required number of solutions.

In our implementation the neighborhood size is set at 30, the tabu list contains 6 elements, and the stopping criterion is set at 3000 visited solutions.

A TS algorithm used in this paper can be outlined as follows.

Let *qsize* be a required size of neighborhood $Q'(S)$, and F'_{best} be the objective function value for the best solution in $Q'(S)$, S'_{best}.

1. Initialize an empty tabu list.
2. Get the initial basic solution S.

3. Repeat until the stopping criterion is satisfied:
   3.1. Set F'_{best} at a great number, and set $counter = 0$.
   3.2. Repeat until $counter = qsize$:
      3.2.1. Generate a neighbor S' of S.
      3.2.2. If a move leading from S to S' is not tabu:
         a) Calculate the objective function value for S', F'.
         b) If $F' < F'_{best}$, set $S'_{best} = S'$ and $F'_{best} = F'$.
         c) Set $counter = counter + 1$;
   3.3. Replace S by S'_{best} and update the tabu list.
4. Return the best solution found.

4. Illustrative example

To illustrate the problem and the solution method we present the following example. Consider the case of the two-stage flowshop with 3 machines at stage 1 and a single machine in stage 2. The number of jobs $n = 15$, the number of resource types $l = 3$, the availability at any moment of a resource of each type, $W_r = 1$. Job processing times and resource requirements are shown in Table 2.

Table 2. Data for the illustrative example

		jobs														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
job processing times at stage 1	machine 1	20	8	17	20	14	16	3	8	10	14	20	6	12	19	18
	machine 2	20	2	2	19	18	8	12	2	7	13	3	19	14	4	17
	machine 3	12	10	2	20	5	10	15	16	14	11	3	18	17	16	6
job processing times at stage 2	machine 1	3	2	1	6	5	2	1	4	4	3	3	6	2	1	5
resource requirements at stage 1	machine 1	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1
	machine 2	0	1	1	1	0	0	1	1	0	0	0	0	1	1	0
	machine 3	1	0	0	0	0	1	0	0	1	0	1	0	0	0	0

resource availability for each resource type is equal to 1

Fig. 1 presents two schedules in the flowshop. Fig. 1a shows the schedule obtained by the LPSR algorithm with the 3rd selection rule (algorithm A3). Fig. 1b presents the schedule obtained by algorithm A3T. We can see that the schedule produced by algorithm A3T has significantly smaller makespan than the one created by algorithm A3.

The first stage schedules in Figs. 1a and 1b consist of 14 partial schedules. In each of the partial schedules, at most 3 jobs are processed simultaneously and the total resource usage does not exceed the resource availability, W_r=1 for each resource type. For example, in the partial schedule of index 2, jobs 9,

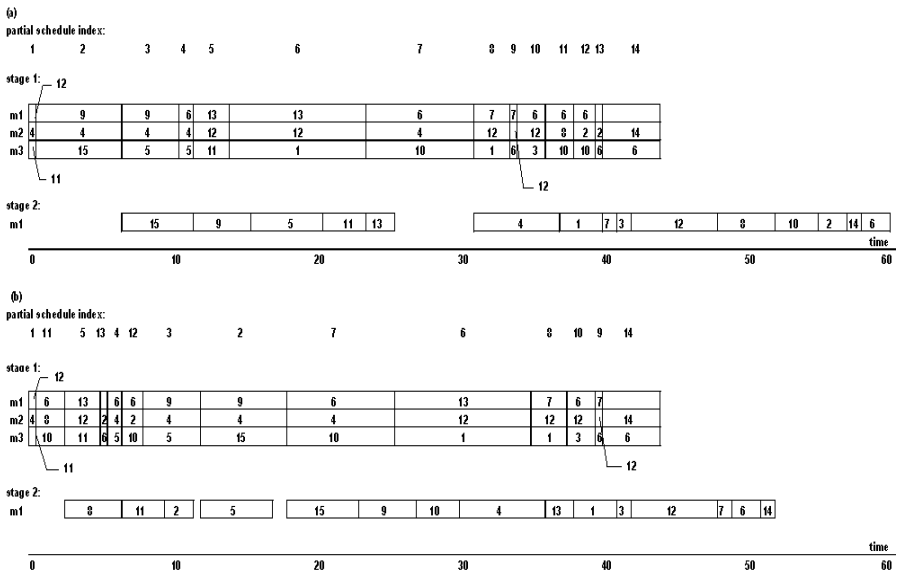

Figure 1. Schedules in the flowshop created by (a) A3 and (b) A3T

4 and 15 are processed simultaneously and use at every moment resources of types 3, 2 and 1, respectively. The jobs in successive partial schedules in Fig. 1a have been selected by the 3rd selection rule. In Fig. 1b the partial schedules are the same as in Fig. 1a, but their ordering is different.

5. Computational study

In this section, the results of a computational experiment conducted to evaluate the performance of the proposed algorithms are presented.

To evaluate the quality of the heuristic solutions we use the values of the percentage deviation of the heuristic makespan from the lower bound on the optimal makespan:

$$\delta = \frac{C_{\max} - LB}{LB} \times 100\%,$$

where $C_{\max}$ is the best makespan found by the heuristic algorithm, and LB is the lower bound on the optimal makespan. $LB = \max\{LB_1, LB_2\}$ where $LB_1 = \sum_{j=1}^{n} s_j + \min_{i=1,\ldots,m,j=1,\ldots,n}\{p_{ij}\}$ and $LB_2 = C_1^* + \min_{j=1,\ldots,n}\{s_j\}$. The second term in LB_1 is due to the fact that the machine at stage 2 remains idle for at least the time needed to finish the processing a job with the smallest processing time at stage 1. The second term in LB_2 results from the fact that after finishing processing all the jobs at stage 1 at least one job with the shortest

processing time must be executed at stage 2. C_1^* denotes the optimal length of the schedule at stage 1 ($C_1^* = T^*$).

The number of jobs n was equal to 40, 80, and 120, the number of machines at stage 1, m , was set at 4 and 8.

The number of resources types l was considered to be m and $m/2$, and the resource availability W_r was equal to 1 and 2 for $l = m$ and $l = m/2$, respectively.

The processing times of jobs at stage 1 were generated from $U[\pi, 2\pi]$ ($U[a, b]$ denotes the discrete uniform distribution in the range of $[a, b]$), where π was set at 20, 30, 40, 50, 60, 70, 80, 90, 100 and 110 for $m = 4$, and π was equal to 30, 50, 70, 90, 110, 130, 150, 170 and 190 for $m = 8$. The processing times at stage 2 were generated from $U[10, 20]$. To compare the results obtained for different intervals of processing times we introduced the stage dominance factor denoted by θ and defined as the ratio of the optimal length of the schedule at stage 1 to the sum of the job processing times at stage 2, $\theta = C_1^* / \sum_j s_j$. The considered problems include three ranges of θ: (i) θ is greater than 1, where stage 1 is dominant ($C_1^* > \sum_j s_j$); (ii) θ is less than 1, where stage 2 is dominant ($\sum_j s_j > C_1^*$); (iii) θ is close to 1, where stages 1 and 2 are balanced ($C_1^* \cong \sum_j s_j$).

For each problem size and for each processing time interval, 50 instances were generated. Thus, 5700 randomly generated instances were created and examined.

All programs for the algorithms presented here were written in C++ and run on a PC with Celeron 2.4 GHz Processor. The LP problems were solved using lp_solve optimizer v.5.5 from `http://groups.yahoo.com/group/lp_solve`.

The results of the computational experiment are presented in Tables 3-5 and in Figs. 2-3. In Tables 3-4, the average percentage deviations of the heuristic makespan from the lower bound on the optimal makespan are shown for algorithms A1-A1T, A2-A2T, A3-A3T, A4-A4T, A5-A5T. In these tables, the average values of dominance factors θ calculated for each problem size and each processing time interval are also depicted (column 4). The results presented in Table 3 were obtained for problems with the number of resource types l equal to m and resource availability W_r equal to 1, the results in Table 4 were obtained for problems with $l = m/2$ and $W_r = 2$ ($r = 1, \ldots, l$). Table 5 contains the computation times of the algorithms. Fig. 2 shows graphically the relationship between the values of θ and the effectiveness of algorithms (A5, A5G, A5S and A5T) based on the instances with 80 jobs and m and l equal to 8. Fig. 3 shows the average values of δ over the entire collection of instances for the problems with the number of resource types l equal to m and $m/2$.

In Tables 3-4 we can see that for all the sizes of problems, the performance of all the algorithms changes in a similar way with changing value of θ. Namely, for all the algorithms, deviations δ are very small when $\theta < 0.9$, they have maximal values for θ close to 1, and decrease with increasing θ for $\theta > 1$. The integration of algorithms A1-A5 with metaheuristics GA, SA and TS results in

the increase of the effectiveness of these algorithms.

In most cases GA and TS provide almost the same improvement and SA produces slightly worse results than GA and TS. Although GA and TS produce solutions of similar quality, there are some differences in their performance for different values of θ. Namely, TS is usually superior to GA in the regions of θ close to 1, while in other regions of θ GA is slightly better than TS.

The improvement of the effectiveness of the algorithms owing to the metaheuristics is the biggest when θ is close to 1 (except for few cases with $\theta < 1$, where the improvement is yet bigger). The significant improvement of results in this region of θ is important from the practical point of view due to the small idle time in both stages (the stages are balanced, i.e. the length of the optimal schedule in stage 1 is close to the minimal length of the schedule in stage 2). In most cases (in Table 3), the effectiveness of algorithms A1T- A5T is more than two times better than, respectively, algorithms A1-A5 when θ has the values closest to 1 (e.g. for the problems with ($n = 40$, $m = 4$, $\theta = 1.03$), ($n = 40$, $m = 8$, $\theta = 1.07$), ($n = 80$, $m = 4$, $\theta = 0.99$)).

The values of δ in Table 4 are slightly higher than those in Table 3. This can be explained by the fact that in case of $l = m/2$ and $W_r = 2$ the resource constraints are weaker than for problems with $l = m$ and $W_r = 1$ and that the performance of the considered algorithms improves with the increasing strength of the resource constraints.

Fig. 2 shows graphically how the metaheuristics GA, AS and TS improve the results. In this figure, the values of deviation δ provided by algorithms A5, A5G, A5S, A5T for problems with $n = 80$, $m = l = 8$ are shown in terms of θ. Other algorithms behave in a similar way for all the considered problems.

Fig. 3 shows the values of δ averaged over all the considered instances for all the considered algorithms in the cases of the numbers of resource types equal to m and $m/2$.

Table 5 shows the average computation times over all the processing time intervals for all the algorithms (computation times were almost the same for all the processing time intervals). In this table, we can see that the computation times of the algorithms increase with the number of jobs, especially the computation times of the algorithms using GA increase considerably when the number of jobs grows. The number of machines and the number of resource types do not have any significant effect on the computation times of the algorithms.

6. Summary and concluding remarks

In this paper, we proposed a heuristic, which integrates various metaheuristics with the linear programming based algorithms for solving the minimal makespan problem of preemptive scheduling in the two-stage flowshop with unrelated machines and renewable resources constraints in the first stage and a single machine in the second stage. First, the heuristic creates a sequence of partial schedules in which jobs are assigned to machines for simultaneous processing during some

Table 3. The average percentage deviations δ for various ranges $[\pi, 2\pi]$ of job processing times and the average values of θ for problems with the number of resource types equal to m

				δ																			
n	m	π	θ	A1	A1G	A1S	A1T	A2	A2G	A2S	A2T	A3	A3G	A3S	A3T	A4	A4G	A4S	A4T	A5	A5G	A5S	A5T
40	4	20	0.54	0.37	0.11	0.12	0.15	0.01	0.00	0.00	0.00	0.15	0.07	0.09	0.08	0.04	0.02	0.02	0.03	0.12	0.05	0.07	0.07
		30	0.80	1.07	0.39	0.30	0.38	0.43	0.06	0.06	0.06	0.35	0.19	0.23	0.21	0.15	0.07	0.08	0.08	0.35	0.19	0.21	0.21
		40	1.03	3.89	2.13	2.88	1.56	2.26	0.91	1.14	0.81	1.49	0.77	0.95	0.68	1.52	0.61	0.73	0.60	1.49	0.75	0.97	0.68
		50	1.36	3.67	1.85	1.91	1.75	3.14	1.65	1.63	1.61	2.07	1.19	1.22	1.19	2.24	1.27	1.35	1.28	2.07	1.20	1.28	1.19
		60	1.62	3.64	1.78	1.81	1.69	3.30	1.77	1.80	1.82	1.99	1.13	1.18	1.14	2.34	1.38	1.42	1.39	1.99	1.13	1.16	1.14
		70	1.86	3.14	1.50	1.59	1.50	2.63	1.47	1.50	1.50	1.68	1.07	1.08	1.07	1.95	1.18	1.18	1.18	1.68	1.07	1.09	1.08
		80	2.15	2.50	1.17	1.21	1.21	2.02	1.13	1.14	1.14	1.40	0.80	0.81	0.81	1.52	0.89	0.93	0.91	1.40	0.80	0.84	0.80
		90	2.39	1.87	1.04	1.08	1.05	1.79	0.97	1.00	1.02	1.02	0.68	0.68	0.68	1.42	0.78	0.80	0.78	1.02	0.68	0.68	0.68
		100	2.62	1.91	0.98	1.02	0.98	1.78	1.03	1.04	1.05	1.10	0.74	0.75	0.75	1.27	0.83	0.83	0.83	1.10	0.74	0.76	0.75
		110	2.93	1.57	0.84	0.87	0.87	1.41	0.85	0.86	0.86	0.92	0.59	0.60	0.60	1.00	0.62	0.63	0.62	0.92	0.59	0.59	0.59
	8	30	0.45	0.13	0.05	0.06	0.08	0.02	0.01	0.01	0.01	0.14	0.08	0.08	0.10	0.05	0.03	0.04	0.05	0.14	0.07	0.08	0.10
		50	0.77	0.41	0.27	0.26	0.28	0.26	0.14	0.13	0.14	0.44	0.28	0.32	0.28	0.36	0.18	0.18	0.19	0.44	0.28	0.37	0.28
		70	1.07	4.33	2.71	3.61	2.07	2.59	1.24	1.76	1.16	2.23	1.17	1.40	0.98	1.91	0.87	1.04	0.80	2.23	1.06	1.47	1.05
		90	1.38	7.65	5.34	5.76	4.93	7.02	4.45	4.96	4.27	5.11	3.31	3.91	3.23	5.55	3.63	4.01	3.59	5.11	3.33	3.71	3.23
		110	1.78	4.54	2.97	3.29	3.00	4.30	2.92	3.04	2.94	3.10	2.34	2.44	2.32	3.37	2.49	2.51	2.52	3.10	2.31	2.42	2.35
		130	2.02	4.77	2.96	3.10	2.98	4.28	2.75	2.91	2.81	3.29	2.20	2.22	2.21	3.47	2.40	2.49	2.41	3.29	2.17	2.29	2.20
		150	2.28	4.08	2.64	2.77	2.67	3.98	2.52	2.65	2.61	2.85	1.98	2.11	1.99	3.07	2.07	2.12	2.13	2.85	1.98	2.01	1.98
		170	2.64	2.93	1.90	1.96	1.93	2.80	1.91	1.96	1.93	2.05	1.47	1.51	1.47	2.33	1.57	1.59	1.61	2.05	1.47	1.51	1.49
		190	2.96	2.86	1.79	1.84	1.81	2.61	1.81	1.86	1.82	2.17	1.42	1.51	1.42	2.27	1.54	1.59	1.56	2.17	1.42	1.49	1.42
80	4	20	0.49	0.09	0.03	0.07	0.05	0.00	0.00	0.00	0.00	0.09	0.06	0.07	0.07	0.03	0.00	0.01	0.01	0.06	0.05	0.06	0.05
		30	0.74	0.26	0.11	0.17	0.12	0.03	0.00	0.01	0.00	0.21	0.11	0.12	0.13	0.06	0.01	0.01	0.01	0.21	0.11	0.13	0.13
		40	0.99	2.08	1.42	1.71	0.87	1.00	0.46	0.58	0.41	0.72	0.40	0.48	0.35	0.71	0.33	0.38	0.39	0.72	0.37	0.47	0.36
		50	1.23	2.82	1.88	2.11	1.44	2.00	1.06	1.28	1.07	1.07	0.59	0.76	0.58	1.32	0.68	0.83	0.71	1.07	0.58	0.74	0.59
		60	1.44	2.40	1.48	1.53	1.25	1.88	1.09	1.12	1.09	1.23	0.66	0.74	0.68	1.35	0.77	0.83	0.82	1.23	0.68	0.69	0.69
		70	1.74	1.50	0.78	0.84	0.83	1.41	0.76	0.82	0.84	0.90	0.47	0.49	0.48	1.02	0.54	0.57	0.55	0.90	0.46	0.50	0.51
		80	1.95	1.22	0.61	0.64	0.63	1.26	0.61	0.66	0.64	0.70	0.38	0.40	0.38	0.80	0.44	0.46	0.46	0.70	0.37	0.38	0.38
		90	2.26	1.18	0.52	0.58	0.58	1.14	0.54	0.58	0.59	0.64	0.34	0.35	0.34	0.75	0.38	0.41	0.41	0.64	0.34	0.36	0.34
		100	2.43	1.05	0.56	0.62	0.61	1.05	0.57	0.59	0.60	0.59	0.36	0.36	0.36	0.69	0.39	0.40	0.40	0.59	0.36	0.37	0.36
		110	2.77	0.85	0.42	0.46	0.46	0.89	0.44	0.48	0.47	0.49	0.30	0.31	0.32	0.63	0.34	0.35	0.36	0.49	0.30	0.30	0.32

Table 4. Table 3. cont.

				δ																			
n	m	π	θ	A1	A1G	A1S	A1T	A2	A2G	A2S	A2T	A3	A3G	A3S	A3T	A4	A4G	A4S	A4T	A5	A5G	A5S	A5T
80	8	30	0.42	0.06	0.03	0.05	0.04	0.00	0.00	0.00	0.00	0.06	0.04	0.05	0.04	0.01	0.00	0.01	0.01	0.06	0.04	0.05	0.04
		50	0.68	0.20	0.08	0.13	0.07	0.01	0.00	0.00	0.00	0.14	0.10	0.11	0.10	0.04	0.01	0.02	0.02	0.14	0.09	0.13	0.11
		70	0.97	1.77	1.35	1.59	0.89	1.17	0.76	0.88	0.60	0.93	0.60	0.70	0.45	0.94	0.48	0.55	0.45	0.93	0.53	0.70	0.45
		90	1.25	5.06	4.02	4.31	3.30	4.21	2.87	3.17	2.64	2.91	1.91	2.31	1.70	2.95	1.96	2.14	1.83	2.91	1.93	2.21	1.70
		110	1.50	4.25	3.26	3.57	2.92	3.91	2.62	2.88	2.54	2.58	1.72	1.99	1.72	2.90	1.91	2.07	1.84	2.58	1.72	1.93	1.71
		130	1.83	3.27	2.14	2.30	1.98	2.95	1.85	1.99	1.91	1.94	1.27	1.39	1.26	2.18	1.37	1.47	1.42	1.94	1.26	1.37	1.27
		150	2.04	3.24	1.97	2.12	1.93	2.86	1.71	1.89	1.80	2.01	1.25	1.38	1.24	2.04	1.35	1.44	1.42	2.01	1.26	1.40	1.27
		170	2.35	2.54	1.44	1.63	1.48	2.31	1.36	1.46	1.39	1.58	1.00	1.04	1.03	1.68	1.10	1.12	1.15	1.58	1.00	1.05	1.01
		190	2.55	2.14	1.26	1.38	1.32	2.00	1.27	1.36	1.32	1.41	0.86	0.91	0.90	1.51	0.96	1.05	1.00	1.41	0.86	0.92	0.88
120	4	20	0.47	0.07	0.04	0.06	0.05	0.00	0.00	0.00	0.00	0.08	0.03	0.05	0.04	0.00	0.00	0.00	0.00	0.04	0.02	0.04	0.03
		30	0.71	0.15	0.09	0.11	0.06	0.00	0.00	0.00	0.00	0.11	0.04	0.08	0.05	0.01	0.01	0.01	0.01	0.11	0.05	0.10	0.07
		40	0.96	1.23	0.96	1.16	0.70	0.29	0.15	0.14	0.13	0.33	0.14	0.18	0.14	0.21	0.09	0.12	0.08	0.33	0.16	0.24	0.14
		50	1.19	2.32	1.76	1.83	1.31	1.81	0.96	1.14	0.91	0.92	0.56	0.65	0.57	1.12	0.63	0.68	0.64	0.92	0.56	0.65	0.57
		60	1.45	1.64	0.81	0.80	0.75	1.26	0.69	0.75	0.76	0.77	0.44	0.48	0.45	0.91	0.50	0.57	0.54	0.77	0.44	0.48	0.44
		70	1.68	1.18	0.52	0.61	0.59	1.06	0.52	0.57	0.58	0.60	0.33	0.35	0.35	0.67	0.37	0.43	0.42	0.60	0.33	0.35	0.35
		80	1.93	1.19	0.49	0.56	0.57	0.98	0.46	0.54	0.54	0.61	0.32	0.33	0.34	0.69	0.36	0.41	0.41	0.61	0.32	0.33	0.34
		90	2.17	0.81	0.36	0.41	0.42	0.89	0.41	0.45	0.46	0.53	0.25	0.27	0.26	0.61	0.30	0.33	0.34	0.53	0.25	0.30	0.27
		100	2.38	0.65	0.30	0.34	0.35	0.61	0.31	0.35	0.35	0.36	0.21	0.22	0.21	0.42	0.22	0.24	0.25	0.36	0.21	0.22	0.21
		110	2.61	0.82	0.38	0.45	0.42	0.78	0.39	0.44	0.43	0.45	0.26	0.28	0.27	0.59	0.29	0.32	0.33	0.45	0.26	0.28	0.27
	8	30	0.38	0.03	0.02	0.03	0.02	0.00	0.00	0.00	0.00	0.03	0.02	0.03	0.02	0.01	0.00	0.00	0.01	0.03	0.02	0.03	0.02
		50	0.64	0.08	0.05	0.04	0.03	0.00	0.00	0.00	0.00	0.04	0.03	0.04	0.04	0.02	0.01	0.01	0.01	0.04	0.04	0.04	0.04
		70	0.89	0.95	0.83	0.92	0.54	0.24	0.08	0.18	0.07	0.25	0.15	0.20	0.13	0.15	0.09	0.11	0.06	0.25	0.15	0.18	0.15
		90	1.15	4.47	4.02	4.37	3.36	3.00	2.39	2.64	1.98	1.98	1.54	1.79	1.26	2.09	1.55	1.60	1.39	1.98	1.48	1.79	1.27
		110	1.40	3.59	2.95	3.12	2.44	2.86	2.02	2.13	1.96	1.88	1.33	1.46	1.26	2.17	1.42	1.70	1.47	1.88	1.36	1.52	1.28
		130	1.70	2.65	1.94	2.14	1.83	2.46	1.58	1.76	1.64	1.61	1.09	1.15	1.05	1.82	1.15	1.26	1.17	1.61	1.08	1.19	1.07
		150	1.89	2.73	1.71	1.90	1.63	2.49	1.47	1.57	1.54	1.53	1.01	1.13	1.00	1.69	1.07	1.13	1.06	1.53	0.97	1.02	1.00
		170	2.19	2.03	1.26	1.32	1.24	1.92	1.10	1.21	1.19	1.33	0.81	0.84	0.83	1.40	0.86	0.91	0.91	1.33	0.79	0.86	0.82
		190	2.41	1.92	1.18	1.24	1.21	1.80	1.04	1.16	1.15	1.18	0.73	0.78	0.77	1.33	0.80	0.85	0.84	1.18	0.73	0.78	0.77

Table 5. The average percentage deviations δ for various ranges $[\pi, 2\pi]$ of job processing times and the average values of θ for problems with the number of resource types equal to $m/2$

				δ																			
n	m	π	θ	A1	A1G	A1S	A1T	A2	A2G	A2S	A2T	A3	A3G	A3S	A3T	A4	A4G	A4S	A4T	A5	A5G	A5S	A5T
40	4	20	0.45	0.19	0.08	0.12	0.11	0.00	0.00	0.00	0.00	0.16	0.05	0.10	0.07	0.03	0.01	0.02	0.01	0.14	0.05	0.08	0.07
		30	0.68	0.28	0.15	0.15	0.17	0.04	0.00	0.00	0.00	0.32	0.17	0.26	0.24	0.11	0.06	0.07	0.08	0.32	0.17	0.30	0.23
		40	0.91	2.38	1.40	1.84	1.14	0.68	0.32	0.41	0.30	0.73	0.45	0.53	0.43	0.39	0.27	0.30	0.30	0.73	0.41	0.51	0.42
		50	1.12	6.02	4.73	5.36	3.97	3.85	2.83	3.05	2.62	2.09	1.62	1.80	1.58	2.31	1.70	1.95	1.68	2.09	1.59	1.84	1.59
		60	1.33	5.98	4.27	4.57	3.96	5.16	3.83	3.91	3.82	3.39	2.56	2.71	2.61	3.44	2.76	2.86	2.80	3.39	2.55	2.71	2.53
		70	1.55	4.69	2.92	3.18	2.96	4.10	2.86	2.90	2.88	2.50	1.97	2.00	1.97	2.94	2.18	2.22	2.18	2.50	1.97	2.01	1.97
		80	1.84	4.00	2.36	2.43	2.38	3.70	2.47	2.48	2.52	2.16	1.66	1.69	1.67	2.54	1.86	1.88	1.86	2.16	1.66	1.66	1.66
		90	2.05	3.46	2.18	2.21	2.20	3.16	2.15	2.24	2.16	2.07	1.55	1.57	1.56	2.25	1.69	1.70	1.70	2.07	1.55	1.57	1.56
		100	2.24	2.91	1.94	2.00	1.96	2.77	2.01	2.04	2.08	1.76	1.38	1.40	1.39	1.88	1.51	1.52	1.51	1.76	1.38	1.41	1.39
		110	2.49	2.29	1.58	1.64	1.63	2.26	1.66	1.67	1.68	1.48	1.16	1.16	1.16	1.61	1.25	1.27	1.27	1.48	1.16	1.16	1.16
	8	30	0.37	0.09	0.03	0.02	0.04	0.00	0.00	0.00	0.00	0.08	0.03	0.05	0.03	0.01	0.01	0.01	0.01	0.08	0.03	0.06	0.03
		50	0.62	0.16	0.11	0.13	0.14	0.03	0.02	0.02	0.01	0.16	0.12	0.14	0.12	0.06	0.05	0.05	0.06	0.16	0.11	0.13	0.12
		70	0.86	1.08	0.63	0.73	0.60	0.36	0.18	0.18	0.19	0.39	0.33	0.38	0.34	0.41	0.28	0.29	0.27	0.39	0.32	0.38	0.34
		90	1.12	7.35	5.51	6.37	4.76	4.84	3.07	3.89	3.02	3.10	2.27	2.70	2.18	3.28	2.30	2.56	2.29	3.10	2.22	2.72	2.30
		110	1.34	9.69	7.45	8.54	7.12	7.84	6.40	6.96	6.36	6.05	4.76	5.13	4.79	6.08	5.01	5.27	4.89	6.05	4.78	5.06	4.73
		130	1.59	6.72	5.29	5.62	5.10	6.67	5.09	5.37	5.06	4.98	3.83	4.07	3.86	5.13	4.06	4.24	4.08	4.98	3.82	4.01	3.84
		150	1.79	6.87	5.33	5.72	5.24	6.52	5.14	5.53	5.18	5.08	3.87	4.06	3.89	5.25	4.17	4.37	4.18	5.08	3.86	4.12	3.89
		170	2.07	5.69	4.29	4.61	4.30	5.54	4.13	4.31	4.23	4.05	3.15	3.23	3.15	4.13	3.27	3.32	3.26	4.05	3.16	3.30	3.15
		190	2.31	4.84	3.71	3.89	3.71	4.38	3.58	3.63	3.63	3.48	2.76	2.80	2.76	3.67	2.94	3.08	3.01	3.48	2.76	2.77	2.77
80	4	20	0.44	0.09	0.02	0.06	0.04	0.00	0.00	0.00	0.00	0.05	0.02	0.04	0.04	0.00	0.00	0.00	0.00	0.03	0.02	0.03	0.02
		30	0.66	0.25	0.11	0.13	0.12	0.00	0.00	0.00	0.00	0.14	0.08	0.13	0.09	0.02	0.01	0.02	0.02	0.14	0.08	0.13	0.09
		40	0.86	0.75	0.57	0.74	0.35	0.05	0.02	0.03	0.02	0.28	0.16	0.23	0.18	0.05	0.03	0.05	0.04	0.28	0.16	0.25	0.18
		50	1.08	4.30	3.85	4.09	3.11	1.95	1.43	1.65	1.40	1.20	0.87	1.10	0.87	1.21	0.94	1.02	0.92	1.20	0.85	1.07	0.83
		60	1.32	2.62	1.94	2.06	1.68	2.17	1.45	1.61	1.47	1.25	0.93	0.96	0.92	1.47	1.01	1.07	1.06	1.25	0.95	0.99	0.94
		70	1.52	2.46	1.66	1.71	1.63	2.01	1.42	1.45	1.48	1.20	0.91	0.98	0.92	1.37	1.02	1.07	1.07	1.20	0.91	0.97	0.93
		80	1.74	2.22	1.26	1.39	1.34	1.98	1.18	1.29	1.27	1.12	0.82	0.83	0.85	1.30	0.90	0.96	0.90	1.12	0.81	0.84	0.84
		90	1.94	1.66	1.05	1.14	1.12	1.69	1.07	1.14	1.17	1.00	0.70	0.71	0.72	1.02	0.75	0.78	0.76	1.00	0.70	0.74	0.72
		100	2.13	1.72	1.08	1.12	1.12	1.47	1.05	1.08	1.09	0.85	0.65	0.67	0.65	0.95	0.74	0.76	0.75	0.85	0.65	0.67	0.65
		110	2.40	1.28	0.81	0.86	0.86	1.30	0.86	0.91	0.91	0.72	0.58	0.60	0.59	0.84	0.64	0.65	0.66	0.72	0.58	0.60	0.59

Table 6. Table 4. cont.

				δ																			
n	m	π	θ	A1	A1G	A1S	A1T	A2	A2G	A2S	A2T	A3	A3G	A3S	A3T	A4	A4G	A4S	A4T	A5	A5G	A5S	A5T
80	8	30	0.34	0.03	0.01	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.01	0.01	0.02
		50	0.56	0.10	0.06	0.08	0.06	0.00	0.00	0.00	0.00	0.07	0.06	0.07	0.07	0.02	0.00	0.01	0.00	0.07	0.05	0.07	0.07
		70	0.79	0.23	0.17	0.23	0.13	0.04	0.02	0.02	0.02	0.14	0.12	0.12	0.12	0.05	0.03	0.05	0.04	0.14	0.12	0.14	0.12
		90	1.03	3.39	3.09	3.29	2.52	1.76	1.34	1.66	1.07	1.03	0.75	1.01	0.69	1.11	0.84	1.01	0.77	1.03	0.73	0.94	0.69
		110	1.27	5.31	4.58	5.16	4.24	4.09	3.53	3.78	3.37	2.88	2.40	2.67	2.28	3.08	2.58	2.80	2.47	2.88	2.42	2.71	2.30
		130	1.54	4.09	3.44	3.63	3.23	3.51	2.85	3.14	2.78	2.34	1.98	2.04	1.98	2.65	2.11	2.26	2.13	2.34	1.93	2.15	1.96
		150	1.71	3.59	3.01	3.18	2.94	3.28	2.60	2.83	2.70	2.21	1.84	1.92	1.84	2.45	1.92	2.11	1.98	2.21	1.82	1.87	1.84
		170	2.00	2.76	2.02	2.35	2.03	2.70	1.97	2.11	2.06	1.81	1.37	1.43	1.41	1.93	1.48	1.56	1.48	1.81	1.36	1.44	1.42
		190	2.17	2.90	2.13	2.34	2.14	2.52	1.96	2.11	2.03	1.84	1.40	1.50	1.41	1.91	1.49	1.56	1.51	1.84	1.40	1.50	1.42
120	4	20	0.42	0.06	0.02	0.05	0.03	0.00	0.00	0.00	0.00	0.06	0.02	0.06	0.03	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01
		30	0.65	0.09	0.05	0.08	0.05	0.00	0.00	0.00	0.00	0.11	0.05	0.10	0.07	0.01	0.00	0.01	0.00	0.11	0.05	0.09	0.06
		40	0.85	0.56	0.35	0.55	0.25	0.04	0.00	0.00	0.00	0.11	0.06	0.09	0.06	0.01	0.01	0.01	0.00	0.11	0.06	0.10	0.06
		50	1.05	3.28	3.08	3.27	2.53	1.48	1.16	1.27	1.11	0.84	0.65	0.72	0.58	0.81	0.62	0.71	0.64	0.84	0.64	0.77	0.59
		60	1.28	2.08	1.71	1.84	1.42	1.61	1.10	1.14	1.15	0.87	0.69	0.76	0.74	0.94	0.70	0.78	0.78	0.87	0.68	0.74	0.72
		70	1.52	1.32	0.95	1.02	0.98	1.24	0.88	0.96	0.96	0.69	0.52	0.55	0.53	0.79	0.59	0.62	0.62	0.69	0.52	0.55	0.53
		80	1.71	1.38	0.85	0.91	0.89	1.27	0.81	0.90	0.88	0.68	0.52	0.53	0.55	0.78	0.57	0.60	0.61	0.68	0.52	0.52	0.53
		90	1.92	1.23	0.76	0.81	0.85	1.11	0.74	0.80	0.82	0.69	0.50	0.50	0.53	0.76	0.53	0.55	0.56	0.69	0.49	0.51	0.53
		100	2.11	1.08	0.66	0.75	0.76	0.99	0.67	0.73	0.73	0.58	0.43	0.45	0.44	0.64	0.47	0.49	0.50	0.58	0.43	0.44	0.45
		110	2.37	0.84	0.54	0.59	0.59	0.78	0.53	0.59	0.57	0.47	0.36	0.37	0.37	0.55	0.40	0.42	0.42	0.47	0.36	0.38	0.37
	8	30	0.32	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.01	0.02	0.01
		50	0.54	0.04	0.02	0.04	0.03	0.00	0.00	0.00	0.00	0.03	0.02	0.03	0.03	0.01	0.00	0.01	0.00	0.03	0.02	0.03	0.02
		70	0.76	0.15	0.11	0.15	0.08	0.00	0.00	0.00	0.00	0.07	0.06	0.07	0.06	0.01	0.01	0.01	0.01	0.07	0.06	0.06	0.06
		90	0.99	2.65	2.51	2.62	2.02	1.02	0.87	0.95	0.72	0.56	0.49	0.54	0.45	0.64	0.49	0.56	0.48	0.56	0.50	0.54	0.44
		110	1.22	4.46	4.22	4.35	3.86	3.44	2.93	3.21	2.80	2.21	1.87	2.09	1.80	2.42	2.00	2.28	2.00	2.21	1.84	2.12	1.77
		130	1.40	3.52	3.16	3.44	2.89	3.01	2.50	2.72	2.54	1.91	1.64	1.75	1.63	2.17	1.77	1.99	1.84	1.91	1.63	1.76	1.63
		150	1.67	2.58	2.22	2.36	2.09	2.42	1.89	2.06	1.95	1.57	1.20	1.31	1.25	1.67	1.30	1.39	1.34	1.57	1.22	1.30	1.22
		170	1.87	2.51	1.97	2.24	1.97	2.31	1.75	1.87	1.91	1.49	1.16	1.25	1.25	1.64	1.25	1.40	1.32	1.49	1.16	1.30	1.21
		190	2.08	1.92	1.41	1.54	1.48	1.86	1.41	1.49	1.54	1.18	0.91	0.98	0.94	1.34	1.02	1.08	1.08	1.18	0.91	0.99	0.94

Table 7. Computation times in seconds (average values over intervals of job processing times)

n	m		A1	A1G	A1S	A1T	A2	A2G	A2S	A2T	A3	A3G	A3S	A3T	A4	A4G	A4S	A4T	A5	A5G	A5S	A5T
		$l = m$:																				
40	4		1.33	1.52	1.43	1.41	1.28	1.47	1.37	1.36	1.35	1.55	1.44	1.43	1.33	1.53	1.43	1.42	1.32	1.52	1.42	1.41
	8		1.22	1.42	1.31	1.31	1.16	1.35	1.25	1.25	1.21	1.41	1.31	1.30	1.17	1.37	1.26	1.26	1.21	1.41	1.30	1.30
80	4		2.32	3.11	2.53	2.51	2.26	3.00	2.47	2.45	2.34	3.13	2.55	2.54	2.32	3.09	2.53	2.51	2.37	3.16	2.58	2.57
	8		2.44	3.28	2.68	2.65	2.38	3.17	2.61	2.59	2.44	3.28	2.68	2.66	2.42	3.24	2.65	2.64	2.44	3.28	2.68	2.66
120	4		3.54	5.74	3.96	3.91	3.43	5.46	3.83	3.79	3.58	5.79	4.00	3.96	3.55	5.71	3.96	3.92	3.57	5.78	3.99	3.95
	8		3.80	6.12	4.27	4.20	3.69	5.82	4.13	4.07	3.82	6.14	4.28	4.23	3.78	6.05	4.23	4.18	3.81	6.14	4.27	4.22
		$l = m/2$:																				
40	4		1.17	1.37	1.26	1.25	1.15	1.33	1.23	1.23	1.17	1.36	1.25	1.25	1.17	1.36	1.26	1.26	1.19	1.38	1.27	1.27
	8		1.21	1.42	1.30	1.30	1.17	1.36	1.26	1.26	1.20	1.41	1.30	1.29	1.20	1.40	1.29	1.29	1.21	1.41	1.30	1.30
80	4		2.44	3.24	2.66	2.64	2.35	3.07	2.56	2.54	2.45	3.26	2.68	2.65	2.43	3.22	2.65	2.63	2.45	3.25	2.67	2.65
	8		2.38	3.22	2.62	2.59	2.28	3.02	2.51	2.48	2.42	3.25	2.66	2.64	2.36	3.17	2.60	2.58	2.40	3.23	2.64	2.62
120	4		3.80	6.01	4.22	4.17	3.57	5.53	3.96	3.92	3.78	6.00	4.19	4.15	3.78	5.96	4.20	4.15	3.79	6.01	4.21	4.16
	8		3.75	6.07	4.22	4.15	3.65	5.65	4.09	4.03	3.81	6.14	4.28	4.22	3.76	6.01	4.21	4.16	3.86	6.19	4.32	4.27

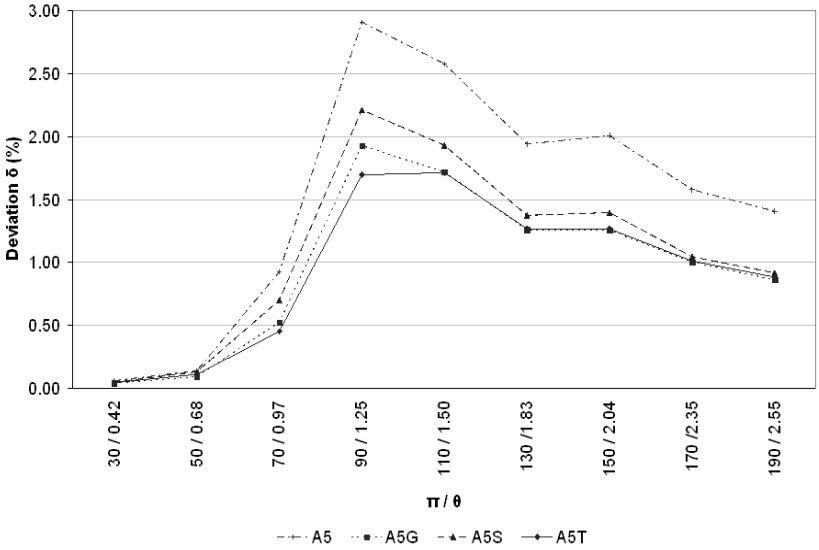

Figure 2. Percentage deviations (obtained by A5, A5G, A5S, A5T) for various ranges $[\pi, 2\pi]$ of job processing times in stage 1 and corresponding average values of θ for problems with $n = 80$ and $=l = 8$

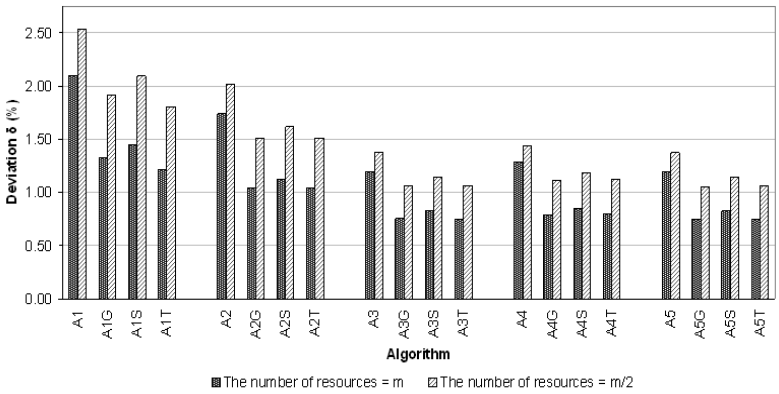

Figure 3. Percentage deviations δ for the number of resources equal to m and $m/2$

periods of time so that the resource constraints are satisfied at every moment. The assignment of the jobs to the machines in successive partial schedules is found by solving the LP problem incorporating a selection rule based on the priority rule. In the second step of the heuristic, one of the three metaheuristics: simulated annealing, tabu search or genetic algorithm, changes the order of the partial schedules so as to improve the results achieved in the first step by five algorithms with five different selection rules.

Performance of the proposed heuristic combining the LP approach with the metaheuristics was experimentally evaluated by comparing the solutions with the lower bound on the optimal makespan. The results show that the greatest improvement in makespan thanks to the use of the metaheuristics has been achieved for the problems in which the optimal length of the schedule in the first stage is close to the sum of job processing times in the second stage. Such problems are the most interesting from the practical point of view due to the shortest total idle time in two stages. The best results have been provided by the tabu search and genetic algorithms, while the simulated annealing algorithm usually produced somewhat worse solutions.

References

Aarts, E.H.L. and van Laarhoven, P.J.M. (1987) *Simulated Annealing: Theory and Applications*. Reidel, Dordrecht.

Adler, L., Fraiman, N., Kobacker, E., Pinedo, M., Plotnicoff, T.P. and Wu, T.P. (1993) A scheduling support system for the packaging industry. *Operations Research* **41**(4), 641-648.

Aldowaisan, T. and Allahverdi, A. (2003) New heuristics for no-wait flowshops to minimize makespan. *Computers & Operations Research* **30**, 1219–1231.

Błażewicz, J., Cellary, W., Słowiński R. and Węglarz, J. (1987) *Scheduling under Resource Constraints: Deterministic Models*. J.C. Baltzer, Basel.

Brah, S.A. (1988) Scheduling in a flow shop with multiple processors. Unpublished Ph.D. Dissertation, University of Huston, Huston, TX.

Brah, S.A. and Loo, L.L. (1999) Heuristics for scheduling in a flow shop with multiple processors. *European Journal of Operational Research* **113**, 113-122.

Chen, B. (1995) Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage. *Journal of the Operational Research Society* **46**, 234-244.

Cheng, T.C.E., Gupta, J.N.D. and Wang, G. (2000) A review of flowshop scheduling research with setup times. *Production and Operations Management* **9**, 262–282.

Crama, Y. (1997) Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Opera-*

tional Research **99**, 136-153.

De Werra, D. (1984) Preemptive scheduling, linear programming and network flows. *placecountry-regionSIAM Journal on Algebraic and Discrete Methods* **5**, 11-20.

De Werra, D. (1988) On the two-phase method for preemptive scheduling. *European. Journal of Operational Research* **37**, 227-235.

Figielska, E. (2008) A new heuristic for scheduling the two-stage flowshop with additional resources. *Computers and Industrial Engineering* **54**, 750-763.

Figielska, E. (2009) A genetic algorithm and a simulated annealing algorithm combined with column generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources. *Computers and Industrial Engineering* **56**, 142-152.

Glover, F. (1989) Tabu search. Part 1. *ORSA Journal of Computing* **1**, 190-206.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Goldberg, D.E., Korb, D. and Deb, K. (1989) Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems* **3**, 493-530.

Gupta, J.N.D. (1988) Two stage hybrid flowshop scheduling problem. *Journal of Operational Research Society* **39**, 359-364.

Gupta, J.N.D. and Stafford, E.F. Jr. (2006) Flowshop scheduling research after five decades. *European Journal of Operational Research* **169**, 699–711.

Haouari, M. and M'Hallah, R. (1997) Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations Research Letters* **21**, 43-53.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Hoogeveen, J.A., Lenstra, J.K. and Veltman, B. (1996) Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard. *European Journal of Operational Research* **89**, 172-175.

Jin, Z., Yang, Z. and Ito, T. (2006) Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economics* **100**, 322–334.

Kellerer, H. and Strusevich, V.A. (2003) Scheduling parallel dedicated machines under a single non-shared resource. *European Journal of Operational Research* **147**, 345-364.

Linn, R. and Zhang, W. (1999) Hybrid flow shop scheduling: a survey. *Computers and Industrial Engineering* **37**, 57-61.

Low, C. (2005) Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers and Operations Research* **32**, 2013–2025.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953) Equation of state calculations by fast computing

machines. *Journal of Chemical Physics* **21**(6), 1087–1092.

Oguz, C. and Ercan, M.F. (2005) A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *Journal of Scheduling* **8**, 323-351.

Salvador, M.S. (1973) A solution of a special class of flowshop scheduling problems. In: S.E. Elmaghraby, ed., *Symposium of the Theory of Scheduling and Its Applications.* Springer-Verlag, New York, 83-91.

Serafini, P. (1996) Scheduling jobs on several machines with the job splitting property. *Operations Research* **44** (4), 617-628.

Słowiński, R. (1980) Two approaches to problems of resource allocation among project activities – A comparative study. *Journal of Operational Research Society* **31**, 711-723.

Słowiński, R. (1981) L'ordonanacement des taches preemptives sur les processeurs independants en presence de ressources supplementaires. *RAIRO Inform./Comp. Science,* **15**, 2, 155-166.

Suresh, V. (1997) A note on scheduling of two-stage flow shop with multiple processors. *International Journal of Production Economics* **49**, 77-82

Vignier, A., Billaut, J.C. and Proust, C. (1999) Les problemes d'ordonnancement de type flow-shop hybride: État de l'art. *RAIRO Recherche opérationnelle* **33**(2), 117–83.

Wardono, B. and Fathi, Y. (2004) A tabu search algorithm for the multistage parallel machine problem with limited buffer capacities. *European Journal of Operational Research* **155**, 380–401.