Raport Badawczy

Research Report

RB/19/2004

Application of artificial intelligence based heuristic algorithms for solving the task scheduling problem

J. Józefczyk

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 8373578

fax: (+48) (22) 8372772

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Zdzisław Bubnicki

Warszawa 2004

scheduling, genetic algorithms, simulation,
artificial intelligence, operations research

Jerzy JÓZEFCZYK*

APPLICATION OF ARTIFICIAL INTELLIGENCE BASED HEURISTIC ALGORITHMS FOR SOLVING THE TASK SCHEDULING PROBLEM

New heuristic algorithms for solving the task scheduling problem with moving executors to minimize the sum of completion times are considered. The corresponding combinatorial optimization problem is formulated for single executor. Hybrid solution algorithms are introduced and investigated, where evolutionary as well as simulated annealing procedures are applied. A simulated annealing algorithm assists an evolutionary algorithm in three different ways. It is used for the generation of the initial set of solutions of the evolutionary algorithm. Moreover, this algorithm attempts to improve the best solutions at current iterations of the evolutionary procedure. The results of the evaluation of the solution algorithms, which have been performed during the computer simulation experiments, are presented. The influence of the parameters of the solution algorithm as well as the task scheduling problem on the quality of results and on the time of computation is investigated.

1. INTRODUCTION

Solving NP–hard combinatorial optimization problems requires searching new more effective approximate and heuristic solution algorithms. Artificial intelligence (AI) based

* Systems Research Institute of Polish Academy of Sciences, Laboratory of Knowledge Systems and Artificial Intelligence, Podwale St. 75, 50–449 Wrocław, Poland

methods are good tools for the determination of heuristic algorithms. In the paper the evolutionary approach has been applied for solving the task scheduling problem with moving executors. This complex discrete decision making problem has been introduced in [3] and other important results are given e.g. in [4,5,6,1], where versions for different scheduling performance indices are investigated. Problems of task scheduling with moving executors are the generalization of the traditional task scheduling problems. It is assumed that *executors* of *tasks* move between places at the plane or in space, where the tasks can be performed. Such places called *workstations* are supplemented by the place called a depot, where executors begin and end their work. Modern discrete manufacturing systems are the main area of applications of this complex scheduling problem. The application of the evolutionary approach to the scheduling problem considered has been investigated in a numerous of papers, e.g. [6,10,8]. The heuristic algorithms presented in the paper consist in co-operation of the evolutionary approach and the simulated annealing metaheuristics. In the consequence, hybrid solution algorithms were determined and in [10] first results of its evaluation were presented. In Section 2 the task scheduling problem with moving executors is introduced and formulated as the binary optimization problem. The case with one executor, non-preemptive, independent tasks to minimize the sum of completion times is only investigated. Then in Section 3 the heuristic solution algorithms are presented and results of their experimental evaluation are given in Section 4. The conclusions complete the paper.

2. PROBLEM OF TASK SCHEDULING WITH MOVING EXECUTORS

The task scheduling problem with moving executors can be introduced as follows. We assume that at every workstation only one task is performed on a plant located there. It is no distinction in notation between a set of tasks and a set of workstations. Both are denoted by $\boldsymbol{H} = \{1, 2, ..., H\}$, where H is a number of tasks (workstations). The element of $\boldsymbol{H}$, i.e. $h \in \boldsymbol{H}$ is an index of the current task (workstation). Analogously, $\boldsymbol{R}$, R, $r \in \boldsymbol{R}$ are a set of executors, a number of executors and an index of the current executor, respectively. To perform the tasks on plants executors should move among workstations and a depot being the additional workstation, where the executors begin and end their movement. Therefore, $\overline{\boldsymbol{H}} = \boldsymbol{H} \cup \{H+1\}$ is the set of workstations with the depot, where $h = H+1$ denotes the depot. Each task consists of two parts: performance of a job at the workstation and driving-up of the executor towards the workstation from the other workstation belonging to the set $\overline{\boldsymbol{H}}$. In the consequence, for each executor a route should be determined in the form of the Hamilton cycle with the beginning and the end at the depot. The routes are the feasible solution of the problem of the task scheduling with moving executors.

From the variety of different scheduling problems a simple case has been selected for the considerations presented in the paper. We assume that tasks are non-preemptive and independent, i.e. no constraint precedence is imposed in the set of tasks, the executors are parallel and unrelated, the ready times are the same for all executors. The assumption of the movement of executors has the influence on the data for the scheduling problem. The

execution times τ_h, $h = 1, 2, ..., H$ are the main data for all task scheduling problems. For unrelated executors the time τ_h is the vector

$$\tau_h = [\tau_{1,h}, \tau_{2,h}, ..., \tau_{r,h}, ..., \tau_{R,h}]^{\mathrm{T}}, \quad h = 1,2, ..., H, \tag{1}$$

where $\tau_{r,h}$ is the execution time of the task h performed by the executor r. For the case under consideration, i.e. when executors are mobile, each time $\tau_{r,h}$ is the sum of two elements: $\bar{\tau}_{r,h}$ and $\tilde{\tau}_{r,g,h}$ being the time the job h is performed at the workstation h by the executor r and the driving-up time of the executor r to the workstation h from the workstation g, respectively. Thus

$$\tau_{r,h} = \bar{\tau}_{r,h} + \tilde{\tau}_{r,g,h}, \quad r = 1,2, ..., R, \quad h = 1,2, ..., H \quad , g = 1,2, ..., H+1. \tag{2}$$

Because each executor should return to the depot, the driving-up time to the depot from the workstation g is additionally denoted by $\tilde{\tau}_{r,g,H+1}$, g=1,2,...,H. It is evident to assume that $\tilde{\tau}_{r,h,h} = +\infty$, $h = 1,2,..., H+1$. Moreover, let us define the matrix $\tau = [\tau_{r,h}]_{\substack{r=1,2,...,R \\ h=1,2,...,H+1}}$.

For the case of the task scheduling problem under consideration the corresponding discrete optimization problem can be formulated. Let us consider now the special case only with one moving executor ($R = 1$) which consists in the minimization of the sum of completion times. The matrix τ for fixed g, $g = 1, 2, ..., H+1$ contains only one row with elements $\bar{\tau}_h + \tilde{\tau}_{g,h}$. For different g they form a square matrix $\tau' = [\tau'_{g,h}]_{g,h=1,2,...,H+1}$. Using τ' and its multiple values a two-dimensional matrix

$$T = [T_{i,h}]_{\substack{i=1,2,\ldots,I \\ h=1,2,\ldots,H+1}} \tag{3}$$

is introduced, where $I = (H+1)^2$ and $T = [\tau'^{\mathrm{T}}, 2\tau'^{\mathrm{T}}, \ldots, (H+1)\tau'^{\mathrm{T}}]^{\mathrm{T}}$. The elements of T are defined as $T_{i,h} = \left\lceil \frac{i}{H+1} \right\rceil \tau'_{g,h}$, where the symbol $\lceil \cdot \rceil$ denotes the least integer number which is not less than its argument. The number of the ith row of T enables us to determine univocally the workstation which corresponds to the times $\tilde{\tau}_{g,h}$ inserted in this row, i.e.

$$g = i - \left\lfloor \frac{i-1}{H+1} \right\rfloor (H+1), \tag{4}$$

where the symbol $\lfloor \cdot \rfloor$ denotes the greatest integer number which is not greater than its argument. We introduce the decision matrix

$$\beta = [\beta_{i,h}]_{\substack{i=1,2,\ldots,I \\ h=1,2,\ldots,H+1}}, \tag{5}$$

where $\beta_{i,h} = 1(0)$ if task h is performed after driving-up from the workstation g which is determined by (4) (otherwise). The following constraints are imposed on matrix β to guarantee obtaining the feasible solution

$$\sum_{i=1}^{I} \beta_{i,h} = 1, \quad h = 1,2,\ldots,H+1, \tag{6}$$

$$\sum_{g=1}^{H+1} \sum_{h=1}^{H+1} \beta_{m,h} \le 1, \quad i = 1,2,\ldots,I, \quad m = \left\lfloor \frac{i-1}{H+1} \right\rfloor (H+1) + g, \tag{7}$$

$$\beta_{i,h} = 1 \Rightarrow \sum_{g=1}^{H+1} \sum_{l=1}^{H+1} \beta_{j,g} = 1, \quad i > H+1, \quad h = 1,2,..,H+1, \quad j = \left(\left\lfloor \frac{i-1}{H+1} \right\rfloor - 1 \right)(H+1) + l, \tag{8}$$

$$\sum_{i=1}^{(H+1)R}\sum_{h=1}^{H}\beta_{i,h}=0, \tag{9}$$

$$\sum_{h=1}^{H+1}\beta_{(p-1)(H+1)+H+1,h}=0 \Rightarrow \sum_{g=1}^{H+1}\sum_{h=1}^{H+1}\beta_{n,h}=0,\ \ p=H+1,...,2,\ \ n=(p-2)(H+1)+g. \tag{10}$$

If constraints (6) and (7) are fulfilled each task is performed and it is impossible for the executor to perform more than one task at the same moment, respectively. The next constraint makes possible to determine the continuous routes of the executors. Two last constraints assure that the depot is the beginning and the end of the movement of the executors. The sum of completion times can be formulated as follows

$$Q_S(\beta)=\sum_{i=1}^{I}\sum_{h=1}^{H+1}T_{i,h}\beta_{i,h}\,. \tag{11}$$

The scheduling problem with moving executors for the case under consideration consists in the determination of matrix β admissible in the sense of constraints (6)–(10) to minimize the performance index (11) if the set $\overline{H}$ as well as the matrix τ are given.

3. AI BASED HEURISTIC SOLUTION ALGORITHMS

The problem formulated in Section 2 is NP–hard optimization combinatorial problem. The heuristic algorithms based on the evolutionary approach and using the simulated annealing metaheuristics were employed till now as the solution tools [8,9]. In the paper the hybrid solution algorithms are proposed which combine the evolutionary as well as the

simulated annealing heuristic solution algorithms. Both component algorithms are shortly introduced. Then different versions of hybrid algorithms are generally described.

The investigations, which have been conducted till now for the different versions of the task scheduling problems with moving executor, allow us to formulate the conclusion that such optimization problems are very sensitive on changes of the parameters of evolutionary algorithms. The crucial are a ratio of crossover p_1 and a ratio of mutation p_2. These parameters can be the result of an adaptation or can be formed during the learning process, [6]. However, in the paper neither adaptation nor learning have been used. The values of p_1 and p_2 are assumed to be a priori known. The admissible solutions denoted by the matrices β and the constraints (6)–(10) can be represented in the simpler way, more convenient for the evolutionary algorithm. As a matter of fact the route of the executor, being the sequence of workstations (tasks) with the beginning and the end at the depot, should be determined. Let us introduce the vector

$$c = [c(n)]_{n=0,1,\ldots,H+1}, \tag{12}$$

where $c(0) = c(H+1) = H+1$ denotes that the executor starts and ends its work at the depot, and $c(n) = h$, i.e. the task h is performed by the executor as the nth consecutive task. The elements $c(n)$, $n = 1, 2, \ldots, H$ correspond to the elements of the matrix β for which $\beta_{i,c(n)=h} = 1$, where $c(n-1) = i - \left\lfloor \frac{i-1}{H+1} \right\rfloor (H+1)$. Moreover,

$$\beta_{i,c(n)} = 1 \rightarrow (\exists c(n+1) \in \overline{H})(\beta_{k,c(n+1)} = 1), \tag{13}$$

where for index k the equation $c(n) = k - \left\lfloor \frac{k-1}{H+1} \right\rfloor (H+1)$ is fulfilled. For example, the vector $c = [6\ 3\ 5\ 1\ 2\ 4\ 6]$ represents the route 6–3–5–1–2–4–6 for $H = 5$. The mutation of the solution is performed according to the following rule. Two elements of c are generated randomly. Then during the other generation according to the rectangular distribution the number from the interval $[0, 1]$ is derived. If this number is less than p_2 then the elements change their positions.

The known crossover method, e.g. [2], enables us to obtain two children solutions c_1', c_2' from two parent solutions c_1, c_2. It is assumed that the set of solutions for each iteration of the algorithm has the same number of elements denoted by *I*. The sequence of solutions is derived randomly and any two consecutive solutions starting from the first one subject to the crossover procedure with the ratio p_1. If the crossover fails then the parent solutions are inserted into the new population. There are *H*/2 possible crossovers for any population for *I* being the even number. Otherwise $(H-1)/2$ crossovers are performed and the new population is supplemented by the one solution randomly chosen from the previous population.

In the evolutionary algorithm presented the tournament method serves for the selection procedure with the parameter *l* being the number of solutions taking part in the tournament. As the fitness function the performance index (11) is applied. The algorithm is stopped if at the *i*th iteration the following condition is fulfilled

$$\sum_{k=j-\eta_0}^{j} \left| \overline{Q}_{S,k} - \overline{Q}_{S,k-1} \right| \leq \varepsilon, \tag{14}$$

where $\overline{Q}_{S,k} = \min_{i=1,2,\ldots,I} Q_{S,k}^{(i)}$, and $Q_{S,k}^{(i)}$ – value of Q_S for the ith solution at the kth iteration of the algorithm,

ε – given accuracy of the solution,

η_0 – parameter being the number of iterations taken into account when the condition (14) is checked.

Let us introduce the basic notation concerning the simulated annealing approach: N – number of iterations of the algorithm which is used also as the stop condition, n, $n = 1, 2, \ldots, N$ – index of the current iteration of the algorithm, θ – parameter of the algorithm called temperature, β_n – solution in the nth step of the algorithm given in the form of matrix β, β' – the best solution determined by the algorithm. The values of parameters N and θ as well as randomly generated initial solution β_0 are the data for the solution algorithm that can be presented in seven steps:

1. Calculate $Q_S(\beta_0)$ and set $n = 0$, $\beta' = \beta_0$.

2. Find a new solution β_{n+1} in the neighborhood of β_n and calculate $Q_S(\beta_{n+1})$.

3. If $Q_S(\beta_{n+1}) \leq Q_S(\beta_n)$ set $\beta' = \beta_{n+1}$ and go to step 6, otherwise go to the next step.

4. Generate randomly the number d from the interval $[0, 1]$ according to the rectangular distribution.

5. If $\exp[Q_S(\beta_n) - Q_S(\beta_{n+1})]/\theta > d$ set $\beta_{n+1} = \beta_n$, otherwise do not change β_{n+1}.

6. Set $\theta = \dfrac{\theta}{1+\lambda\theta}$, where $\lambda = \dfrac{\theta_{\max} - \theta_{\min}}{N\theta_{\max}\theta_{\min}}$.

7. If $n < N$ set $n = n+1$ and go to step 2, otherwise stop the algorithm with the solution represented by β'.

The new solution in step 2 of the algorithm can be determined in different ways. The method used assumes that randomly generated task from the current solution is replaced with all other tasks from this solution. The replacement, which leads to the most improvement of the performance index, is treated as the new solution. If no improvement is obtained then the replacement of the task is randomly generated.

In a numerous of papers the disadvantages of the evolutionary approach to discrete optimization problems have been reported, e.g. [12]. The main difficulties consist in unification of the population (the set of solutions in the current step of the algorithm). Consequently, the global optimum is nearly unreachably. On the other hand the genetic and the evolutionary solution algorithms, because of their features of the parallelism in computations, are strong tools for the optimization problems and it would be unreasonable to reject them. Therefore, modifications and the development of their crucial advantages seems to be the best way to obtain the useful heuristic solution algorithms based on these approaches.

In the paper the additional improvement of the sets of solutions is proposed via the application of the algorithm based on the simulated annealing approach. Such an effective combination of the algorithms was used for different discrete optimization problems, in particular for scheduling problems, e.g. [13, 11]. In all versions considered and presented in

the paper the simulated annealing algorithm starts from the selected solution taken from the set of all solutions at the current step of the evolutionary algorithm and in the consequence gives a new one. The following cases are considered.

a. The simulated annealing algorithm operates on the set of initial solutions which is generated as for the evolutionary algorithm. For each solution the full simulated annealing algorithm is run. In the consequence the completely new initial set of solutions is obtained. This hybrid algorithm is referred to as INT.

b. For the CRT hybrid algorithm the simulated annealing algorithm is run for every k iterations, where $k > 1$. For each such an iteration all solutions are improved by the simulated annealing algorithm.

c. The third case deals with the application of the simulated annealing algorithm only for the solution given by the evolutionary algorithm. This solution is treated as the starting point for the simulated annealing procedure. The hybrid algorithm being the cascade combination of its elements is called FIN.

Of course all versions can be combined but it is not considered in the paper.

4. SIMULATION EXPERIMENTS

The idea of heuristic solution algorithms requires evaluating them during computer simulation experiments which can test their quality for different parameters of decision problem investigated as well as for important parameters of the algorithms. The important

contribution of the paper is an experimental evaluation of the AI based heuristic algorithms introduced in Section 3 for the problem of task scheduling with moving executors. As the basis for the evaluation the value of Q_S and the time of computation T_S have been used. The following default values of all parameters have been assumed as the result of experiments which were performed: $H=100$, $I=100$, $p_1=0.9$, $p_2=0.1$, $l=2$, $\varepsilon=1$, $\eta_0=2$, $N=10000$, $\theta_{\min}=10$, $\theta_{\max}=15000$. Selected results of experiments are given in Tables 1, 2, 3 and in Figs. 1, 2, 3. All values presented are mean values of 30 independent runs of the algorithm. In Fig. 1 the dependence of Q_S^x on the number of tasks H is presented, where $x \in \{\text{INT, CRT, FIN, SA, EA}\}$, and SA, EA denote simple simulated annealing, evolutionary algorithms, respectively. The results are presented in the relative way, i.e. the EA algorithm is treated as the basis. Namely, the values given in the figures are the corresponding ratios of the other algorithm and EA. The ratios are presented as follows: Q_S^{SA}/Q_S^{EA} – Serie 1, Q_S^{INT}/Q_S^{EA} – Serie 2, Q_S^{CRT}/Q_S^{EA} – Serie 3 and Q_S^{FIN}/Q_S^{EA} – Serie 4. The CRT algorithm gives the best results, up to 13%. The simple SA one is considerably worse, especially for the small sizes of the problem. The FIN algorithm doesn't lead to the improvement of the results in comparison with the EA algorithm. In Tables 1 and 2 the result for the number of solutions I (the size of population) and for the parameter η_0 of the stop condition are presented. The maximum difference is about 1%. However, the values of the parameters are very important from the time of computation point of view. The examples of results are given in Table 3 and in Figs. 2,3 for the CRT version. The dependences are rather expected but taking into account the values of Q_S one

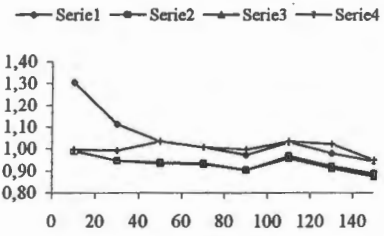

Fig. 1. Dependence of Q_S^x / Q_S^{EA} on H

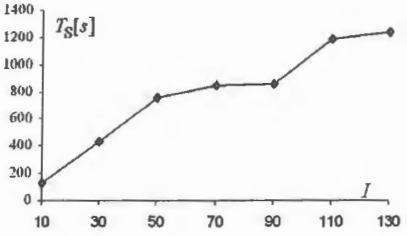

Fig.2. Dependence of $T_S^{CRT}[s]$ on I

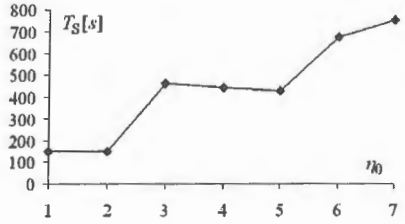

Fig.3. Dependence of $T_S^{CRT}[s]$ on η_0

Table 1. Values of Q_S for different I

10	30	50	70	90	110	130
27449	27339	27298	27183	27219	27139	27164

Table 2. Values of Q_S for different η_0

η_0	1	2	3	4	5	6	7
Q_S	27512	27513	27256	27267	27269	27213	27198

can formulate the following corollaries. It is enough to launch the simulated annealing algorithm in every $k=3$ iteration of the evolutionary algorithm (for the values of parameters assumed the solution was obtained in less than 12 iterations, therefore the greater values of k have not been investigated). The size of the group in the tournament selection procedure has no significant influence on the time of computation. The value of η_0 is directly connected with the number of iterations, therefore the time of computation T_S^{CRT}

Table 3. Values of $T_S^{CRT}[s]$ for different k, ε, I

k			ε			I		
1	2	3	1	2	3	2	3	4
1645	975	293	976	961	956	946	923	987

generally grows but in non-linear form (Fig. 3). Moreover, the slight increasing of the quality of scheduling was noticed (Table 2).

5. CONCLUSIONS

In the paper the solution of the complex task scheduling problem is presented. The movement of executors being the subjects performing the tasks is taken into account. It leads to new discrete optimization problems. Because of the strong computational complexity of these problems the determination of the heuristic algorithms is very important. The proposition of such algorithms based on the AI approaches are given. The versions being the mixture of the evolutionary algorithm and the simulated annealing metaheuristics are considered. To assess the algorithms the computer simulation has been conducted. The dependences of the quality of the scheduling and of the time of computation on the changes of the parameters of the algorithms have been examined. The computer simulations conducted show the significant improvement of the results. When the simulated annealing and the evolutionary algorithms are applied alternatively the results are the best. The improvement up to 13% was obtained in comparison with individual evolutionary algorithm. The results achieved can be treated as the important suggestions for the designing and implementations of the solution algorithm in discrete manufacturing systems.

In further works the results will be compared with the approximate algorithm and for the small instances of the problem with the exact algorithm. Moreover, advanced versions of

the simulated annealing can be used and their usefulness for the hybrid algorithms is worth evaluating.

REFERENCES

[1] AVERBAKH O., BERMAN A., *A simple heuristic for m–machine flow–shop and its applications in routing–scheduling problems*, Operations Research, Vol. 47 (1999), pp. 165–170.

[2] GREFENSTETTE J. *et al*, *Genetic algorithms for the traveling salesman problem*, In: Proceedings of International Conerence on Genetic Algorithms, Lawrence Erlbaum Associates, Mahaw NJ, USA 1985.

[3] JÓZEFCZYK J., *Tasks scheduling on moving executors in complex operation system* (in Polish), Wrocław University of Technology Press, Wrocław 1996.

[4] JÓZEFCZYK J., *An algorithm for scheduling tasks on moving executors in complex operation systems*, In: Proceedings of 1st IFAC Workshop on Manufacturing Systems MIM'97, Wien, Austria 1997, pp. 139–144.

[5] JÓZEFCZYK J. *Scheduling tasks on moving executors to minimise the maximum lateness*, European Journal of Operational Research, Vol. 131 (2001), pp. 171–187.

[6] JÓZEFCZYK J., *Application of genetic algorithms for solving the scheduling problem with moving executors*, Systems Science, Vol. 27 (2001), pp. 87–95.

[7] JÓZEFCZYK J., *Solving of the scheduling problem with moving executors using advanced genetic algorithms*, In: Proceedings of 3rd Symposium AI Meth, Gliwice, Poland 2002, pp. 205–208.

[8] JÓZEFCZYK J., *On the application of evolutionary algorithms with multiple crossovers for solving the task scheduling problem*, In: Proceedings of IASTED International Conference on Intelligent Systems and Control, Salzburg, Austria, Acta Press (Ed. M.H. Hamza) 2003, pp. 301-306.

[9] JÓZEFCZYK J., *Scheduling of tasks on moving executors using advanced evolutionary algorithms*, In: Proceedings of 16th International Conference on Systems Engineering, Vol. 1, Coventry, U.K. 2003, pp. 309-314.

[10] JÓZEFCZYK J., *Hybrid solution algorithms for task scheduling problem with moving executors*, Proceedings of 2nd IFAC Workshop on Advanced Fuzzy/Neural Control, Oulu, Finland 2004 (in press).

[11] LEUNG L., CHAN CH. K., TROUTT D. M., *Application of mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem*, European Journal of Operational Research, Vol. 145 (2003), pp. 530–542.

[12] WEINBERG M., OPPACHER F., *A linear time algorithm for determining population diversity in evolutionary computation*, In: Proceedings of IASTED International Conference on Intelligent Systems and Control, Salzburg, Austria, Acta Press (Ed. M.H. Hamza) 2003, pp. 270-275.

[13] WANG L., DA-ZHONG Z., *An effective hybrid optimization strategy for job-shop scheduling problems*, Computers and Operations Research, Vol. 28, (2001), pp. 585–596.