MISTA 2009

Scatter search for a weighted tardiness flow shop problem

Wojciech Bożejko · Mariusz Uchroński · Mieczysław Wodecki

Extended abstract

This work concerns the artificial intelligence instruments application in the building schedule. In this article an original optimization scatter search algorithm was presented, taking into consideration both technological and organizational restrictions. The algorithm was applied to the real analysis of the industrial building project realization.

The problems of tasks' scheduling with sum-cost goal functions have a long, over thirty years history. A great number of works concern scheduling on one machine. For multi-machine problems research more often involves the issue of delay tasks number minimization, i.e. $\sum U_i$ (where $U_i = 1$, if $C_{i,m} > d_i$, otherwise $U_i = 0$). In the work of Lenstra et al. [5] it is proved that for two machines the problem $F2||\sum U_i$ is strongly NP-hard. To the most important work, containing main theoretical results as well as good approximation algorithms belongs Bulfin and Hallah [1].

For a problem with many machines $F||\sum U_i$ Hairi and Potts [3] described an optimal algorithm based on the division and restriction method, by means of which in a reasonable time instances with 25 tasks and 3 machines can be solved. Genetic algorithms presented in works of Ucar and Tasgetiren [10] are ones of a few metaheuristic algorithms which have been made exclusively to the solve multi-machine flow-shop problem with a criterion $\sum w_i U_i$. There are considerably less works devoted to problems of costs delay sum minimization, i.e. the function $\sum w_i T_i$ and its particular example $\sum T_i$. The difficulty of this problem is verified by the fact that for the simplest example of two machines and goal function $\sum T_i$ the newest exact algorithms (Schaller [8]) solve instances to 20 tasks. Whereas, for the most difficult (discussed in this chapter) problem $F||\sum w_i T_i$ only a few works have been published describing optimal algorithms (the newest one - Chung et al. [2]). A wide range of methods, algorithms and publications covering multi-machine problems of tasks scheduling with costs delay sum minimization is included in works of and Valada et al. [11].

Wojciech Bożejko
Wrocław University of Technology
Institute of Computer Engineering, Control and Robotics
Janiszewskiego 11-17, 50-372 Wrocław, Poland, E-mail: wojciech.bozejko@pwr.wroc.pl

Mariusz Uchroński
Wrocław University of Technology
Institute of Computer Engineering, Control and Robotics
Janiszewskiego 11-17, 50-372 Wrocław, Poland, E-mail: mariusz.uchronski@pwr.wroc.pl

Mieczysław Wodecki
University of Wrocław
Institute of Computer Science
Joliot-Curie 15, 50-383 Wrocław, Poland, E-mail: mwd@ii.uni.wroc.pl

The Total Weighted Tardiness Flow Shop Problem (in brief TWTFS) can be defined in the following way: here is a set of n tasks $\mathcal{J} = \{1, 2, \ldots, n\}$ which are to be performed by means of m machines from the set $\mathcal{P} = \{1, 2, \ldots, m\}$. The task $j \in \mathcal{J}$ should be performed consecutively on each machine, but the task performance on a k machine (operation O_{jk}) can start only after the completion of this task performance on a machine $k-1$ ($k = 2, 3, \ldots, m$) and the sequence of tasks performance on each machine must be the same. Furthermore, in any moment the machine can perform at most one task and the task performance cannot be broken. For the task $i \in \mathcal{J}$, let p_{ij} be the time of performance on the machine $j \in \mathcal{P}$, d_i the required completion time, and w_i the cost function weight.

For the set tasks sequence by $C_{i,j}$ we mark the completion time of the task $i \in \mathcal{J}$ on a machine $j \in \mathcal{P}$. Then, $T_i = \max\{0, C_{i,m} - d_i\}$ is a tardiness, and $f_i(C_{i,m}) = w_i \cdot T_i$ cost of tardiness of task performance. It should be determined such a sequence of tasks performance that it would minimize the costs tardiness sum i.e. the sum $\sum_{i=1}^{n} f_i(C_{i,m}) = \sum_{i=1}^{n} w_i \cdot T_i$.

The scatter-search method. The main ideas of this method are presented also in the work of James et al. [4]. In the proposed algorithm the base of the procedure operation, which generates the path linking two selected solutions is the scheme of multi-step fusion *MSXF* an operator programmed in the work of Reeves and Yamada [7]. As a measure of distance between solutions (permutations) serving to manage the process of making the path between $\pi(i)$ and $\sigma(i)$ the Hamming's measure was adopted: $H(\pi, \sigma) =$ number of i such that $\pi(i) \neq \sigma(i)$. The calculation complexity of determining the distance value is linear $O(n)$, where n is the number of elements in a permutation. In the path-relinking procedure each next path element is determined by looking through the neighborhood, i.e. some subset of solutions space generated by insert moves.

Moves and neighborhood. Let $\pi = (\pi(1), \ldots, \pi(n))$ be a n element permutation of the Φ_n set.

The *insert move* i_l^k presents an element $\pi(k)$ (from the k position in π) on the position l, generating a permutation $i_l^k(\pi) = \pi_l^k$. If $l \geq k$, then

$$\pi_l^k(i) = \begin{cases} \pi(i), & \text{if } i < k \vee i > l, \\ \pi(i+1), & \text{if } k \leq i < l, \\ \pi(k), & \text{if } i = l. \end{cases}$$

and similarly in the case when $k > l$. The insert move we will call in brief *i-move*. Its calculation complexity equals $O(1)$. There are $n(n-1)$ of all such moves.

The permutation neighborhood $\pi \in \Phi_n$ generated by all *i-moves*:

$$\mathcal{N}(\pi) = \{i_l^k(\pi) : \ l, k = 1, 2, ..., n \text{ and } l \neq k\}. \tag{1}$$

The sequences of directly succeeding jobs $\pi^B = (\pi(a), \pi(a+1), \ldots, \pi(b-1), \pi(b))$, $(1 \leq a < b \leq n)$ in permutation $\pi \in \Phi_n$ we call subpermutations. A *weight* of a subpermutation $\mathcal{WT}(\pi^B) = \sum_{i=a}^{b} w_{\pi(i)} \cdot T_{\pi(i)}$, and a *length* $\mathcal{L}(\pi^B) = C_{\pi(b),m}$. Let $\Phi_n(\pi^B) \subseteq \Phi_n$ be a set of permutations which can be generated from a permutation π by swapping an order of elements in a subpermutation π^B. The number of elements of the set $\Phi_n(\pi^B)$ equals $(b-a+1)!$.

Let $\mathcal{F}$ mean one of the defined above goal functions, i.e. $\mathcal{WT}$ (a weight) or $\mathcal{L}$ (a length). For a subpermutation π^B let

$$F^{mi}(\pi^B) \leq \min\{\mathcal{F}(\delta) : \ \delta \in \Phi_n(\pi^B)\},$$

$$F^{mx}(\pi^B) \geq \max\{\mathcal{F}(\delta) :\ \delta \in \Phi_n(\pi^B)\}.$$

For any permutation $\gamma \in \Phi_n(\pi^B)$ a value of $F^{mi}(\pi^B)$ and $F^{mx}(\pi^B)$ are lower and upper bound of $\mathcal{F}(\gamma)$, therefore $\mathcal{F}(\gamma) \in [F^{mi}(\pi^B), F^{mx}(\pi^B)]$. Subpermutations π^B we call $\theta - optimal$ $(0 \leq \theta \leq 1)$ due to the function F, if

$$F(\pi^B) \in [F^{mi}(\pi^B), \theta(F^{mx}(\pi^B) - F^{mi}(\pi^B)].$$

In any permutation we will determine two types of subpermutations called blocks in the further part of the paper: (1) $\mathcal{T}$-blocks, consist of on-time jobs, (2) $\mathcal{D}$-blocks consist of late jobs.

Blocks of on-time jobs. Subpermutation $\pi^{\mathcal{T}} = (\pi(a), \pi(a+1), \ldots \pi(b))$ is a $\mathcal{T}$-***block*** in a permutation π, (a) each job from $\pi^{\mathcal{T}}$ is on-time in the permutation π, so $\forall j \in \{a, a+1, ..., b\},\ C_{\pi(j),m} \leq d_{\pi(j)}$, (b) $\pi^{\mathcal{T}}$ is θ-*optimal* due to the length $\mathcal{L}$. Because each job in a $\mathcal{T}$-block is on-time, therefore the cost of its executing equals 0. Then, $\pi^{\mathcal{T}}$ is an optimal schedule due to the cost.

Blocks of late jobs. Subpermutation of jobs $\pi^{\mathcal{D}} = (\pi(a), \pi(a+1), \ldots \pi(b))$ is a $\mathcal{D}$-***block*** in the permutation $\pi \in \Phi_n$, if: (a$'$) any job $\pi(j)$ $(j = a+1, a+2, \ldots, b)$ moved into the first position in $\pi^{\mathcal{D}}$ (i.e. after executing an *insert* move i_a^j) is late, (b$'$) due to the cost a subpermutation $\pi^{\mathcal{D}}$ is θ-*optimal*.

From the constraint ($\mathbf{a}'$) it follows that each job from a $\mathcal{D}$-block, after moving into the first position in $\pi^{\mathcal{D}}$, is late. Algorithms of blocks determination in a permutation are described in the paper of Wodecki [12]. The number of blocks and the number of elements in each blocks depend on the parameter θ $(0 < \theta < 1)$.

Let $\mathcal{B} = \{\pi^1, \pi^2, ..., \pi^t\}$ be a set of blocks (i.e. $\mathcal{T}$ and $\mathcal{D}$ blocks) in the permutation π and let $\mathcal{N}(\pi)$ be the neighborhood generated by insert moves (see (1)). We are applying restricted neighborhoods in algorithms. We are eliminating permutations generated from π by changing an order of elements in any block. Therefore, we take $\mathcal{N}(\pi) = \mathcal{N}(\pi) \setminus \bigcup_{\delta \in \mathcal{B}} \Phi_n(\pi^\delta)$.

Computational experiments. To conduct the statistical analysis of the algorithm operation the method was also tested on test data. Results were compared with the NEH (Nawaz et al. [6]) algorithm. In Taillard's work [9] there was presented the commonly used these days method of data generation to evaluate approximation algorithms of solution of different combinatorial optimization problems. In their primary version it enables to find instances for the flow shop problem only on the based of the criterion $C_{\max}$. Applying this method the times to conduct the operation were generated. An integer weight was generated from the uniform distribution [1,10]. The required completion times were generated using the following procedure:

Step 1: Calculate $P_i = \sum_{j=1}^{m} p_{i,j},\ \ i = 1, 2, \ldots, n$, the total time of carrying out the task $i \in \mathcal{J}$ on machines.

Step 2: For each task $i \in \mathcal{J}$ determine the required completion time: $d_i = P_i \cdot (1 + \mu \cdot 3)$.

The parameter μ is the random variable realization of a uniform schedule on the [0,1] distance. This generator is described in Taillard's work [9]. The generated data form 8 groups with a size nxm: 20x5 20x10, 50x5, 50x10, 50x20, 100x5, 100x10 and 100x20. For each size 25 examples were determined, so all data embraced 200 examples.

The scatter search algorithm was coded in C++ language. The calculation experiments were performed on the IBM microcomputer with a 1.8 GHz clock. To determine the starting solution the NEH algorithm was employed. In literature, there are neither data nor results to compare with this problem. Therefore, an additional series of instances in a small size nxm was generated: 8x4, 8x6, 10x4, 10x6, 12x4, 12x6 and 12x8 -

10 examples for each size. The obtained results for the data of the NEH algorithm were compared with optimal values determined by a total search. The average relative error equals 8.62%. Before starting the basic test to determine the algorithm parameters calculations were made on a small data group - one example from each size. On this basis it was assumed that the parameter θ (in the block definition) would be equal to 0.2.

Due to the obtained results it can be stated that the blocks application brought about a considerable improvement. The algorithm with blocks determines better solutions in a shorter time. The average improvement of the starting solution (determined by NEH) equals 6.20%; whereas for the algorithm without blocks equals 4.55%. Furthermore, the average time of this algorithm operation is about 22% smaller. The average calculation times of one instance equals 7.25 and 9.13 seconds, respectively. Shortening of the time calculation results from the fact that in case of the algorithm without blocks the whole neighborhood is searched. Number of its elements is $O(n^2)$ and the calculation time of the goal function value for one element is $O(nm)$. In turn, the procedure of blocks determination in a permutation has a calculation complexity $O(n^2m)$. Besides, the calculation experiments proved that restricted neighborhood, which is generated with the use of blocks elimination properties, is about 35% smaller than a whole neighborhood. Therefore, elimination, through filtration (heuristic indirect search) of many non-perspective elements, causes values improvement of fixed solutions and the calculation time shortening.

References

1. Bulfin R.L., Hallah R., Minimizing the weighted number of tardy jobs on two-machineflow shop, Computers&Operations Research, 30, 1887-1900 (2003).
2. Chung C., Flynn J., Kirca O., A branch and bound algorithm to minimize the total tardiness for n-machine permutation flowshop problems, European Journal of Operational Research, 174(1), 1-10 (2006).
3. Hariri A.M.A., Potts C.N., Branch and Bound Algorithm to Minimize the Number of Late Jobs in a Permutation Flowshop, Eur. Jour. of Oper. Res., 38, 228-237 (1989).
4. James T., Rego C., Glover F., Sequential and Parallel Path-Relinking Algorithms for the Quadratic Assignment Problem, IEEE Intelligent Systems, 58-65 (2005).
5. Lenstra J.K., Rinnoy Kan A.G.H., Brucker P., Complexity of Machine Scheduling Problems, Annals of Discrete Mathematics, 1, 343-362 (1977).
6. Nawaz M., Enscore E.E., Ham I., A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, OMEGA, 11/1, 91-95 (1983).
7. Reeves C. R., Yamada T., Solving the Csum Permutation Flowshop Scheduling Problem by Genetic Local Search, IEEE International Conference on Evolutionary Computation, 230-234 (1998).
8. Schaller J., Note on minimizing total tardiness in a two-machine flowshop, Computers&Operations Research, 30(5), 3273-3281 (2005).
9. Taillard E., Benchmarks for basic scheduling problems, European Journal of Operational Research, 64, 278-285 (1993).
10. Ucar H., Tasgetiren M.F., A particle swarm optimization algorithm for permutation flow shop sequencing problem with the number of tardy jobs criterion, Proceedings of 5th International Symposium on Intelligent Manufacturing Systems, 1100-1120 (2006).
11. Valada E., Ruiz R., Minella G., Minimizing Total Tardiness in the m- Machine Flowshop Problem: A Review and Evaluation of Heuristics and Metaheuristics, Computers and Operations Research, Vol. 35, No. 41350-1373 (2006).
12. Wodecki M., A block approach to earliness-tardiness scheduling problems, Advanced Manufacturing Technology, Vol. 40, No. 7-8, 797-807 (2008).