

HEURISTIC APPROACHES TO A COMMON DUE DATE EARLINESS-TARDINESS SCHEDULING PROBLEM

Valery S. Gordon, Alexander A. Tarasevich

National Academy of Sciences of Belarus, United Institute of Informatics Problems, Minsk, Belarus, {gordon@newman.bas-net.by, Tarasevich.Alexander@gmail.com}

Abstract: In this paper we compare simulated annealing and tabu search approaches to a single machine common due date assignment and scheduling problem with jobs available at different times. The objective is to minimize the total weighted sum of earliness, tardiness and due date costs.

1. INTRODUCTION

Among the scheduling problems considered within a “Just-in-Time” (JIT) concept there are problems where jobs are to be completed as close as possible to a common due date. As an example of such problems consider the following problem arising in food industry. Fruits or vegetables from different farms are to be gathered and processed at a plant. If they are delivered too early it is necessary to pay for their storage. Otherwise, if they are processed too late, there would be a loss of money due to a large number of spoiled fruits or vegetables. The situation can be modeled by a following single machine scheduling problem of minimizing earliness-tardiness costs with different ready times of jobs and assignable common due date.

Consider a scheduling situation when n jobs are to be processed by a single machine under common due date d , which is to be assigned. Each job j , $j = 1, 2, \dots, n$, becomes available at moment r_j (*release date, ready time*) and has *processing time* p_j . Let S_j and C_j be *starting* and *completion times* of job j in a certain schedule s . Let $E_j = \max\{0, d - C_j\}$ and $T_j = \max\{0, C_j - d\}$ be *earliness* and *tardiness*

of job j . The objective is to minimize $f(d, s) = \sum_{j=1}^n (\alpha E_j + \beta T_j + \gamma d)$, where $\alpha > 0$, $\beta > 0$ and $\gamma > 0$

are earliness, tardiness and due date per unit penalties. So, we have to find an optimal schedule s^* and an optimal due date d^* to minimize total penalty $f(d, s)$. Note that permitting $\beta = 0$ leads to $d^* = 0$ and to optimality of any feasible schedule. The problem cases with $\alpha = 0$ or $\beta \leq \gamma$ are equivalent to the strongly NP-hard problem of minimizing the sum of completion times subject to given release dates [11], which is well studied. Further we consider the case $\beta > \gamma$. Extending the standard scheme for scheduling notation [10], we refer to our due date assignment problem as $||r_j, d_j := d | \sum (\alpha E_j + \beta T_j + \gamma d)$.

Panwalkar, Smith and Seidmann [12] have considered the case of this problem when all jobs are available at time zero. They show that the optimal common due date coincides with the completion time of the $[K]$ -th job, and provide a polynomial time algorithm for solving the problem. Here $[j]$ denotes a job in position j of job sequence and K is the smallest integer greater or equal to $n(\beta - \gamma)/(\alpha + \beta)$.

Cheng, Chen and Shakhlevich [3] consider the general case of the problem (with non-zero ready times) and proposed a heuristic algorithm. This heuristic consists of two parts. First, determine a sequence σ^* optimal for the relaxed problem with zero ready times by Panwalkar et al. algorithm. Then obtain a schedule s_h^* and a due date d_h^* for the predetermined sequence σ^* of jobs. We refer to this algorithm as to CCS heuristic. The worst case analysis of the CCS algorithm gives the following result [3]: $f(s_h^*, d_h^*) / f(s^*, d^*) \leq 1 + R$. Here s^* and d^* are the optimal schedule and the optimal due date, $R = \max_j \{r_j\}$.

Application of meta-heuristics, namely evolutionary strategies, simulated annealing (SA) and threshold accepting, to a problem with zero ready times, but with $\alpha = \alpha(j)$ and $\beta = \beta(j)$ is considered in [5].

We compare SA and tabu search (TS) approaches to the general case of the problem with constant α and β . Both approaches use the following polynomial-time algorithm [3] proposed for the case of the problem when the sequence of job processing is predetermined.

ALGORITHM 1.

1. Determine starting times of the jobs as follows: $S_{[1]}(\sigma) = r_{[1]}$, $S_{[i]}(\sigma) = \max\{r_{[i]}, C_{[i-1]}(\sigma)\}$.
2. Calculate $u := \lceil n(\beta - \gamma) / \beta \rceil$.
3. Move jobs in positions $1, 2, \dots, u - 1$ to job u so that there will be no idle intervals between them.
4. Calculate $K := \lceil n(\beta - \gamma) / (\alpha + \beta) \rceil$ and set $d_h^* := C_{[K]}$, where $[K]$ is a job in the K -th position of the sequence σ .

The paper is organized as follows. In section 2 we propose a SA approach to the problem and describe corresponding heuristic algorithm. Section 3 contains a TS algorithm description. The computational results on comparison of SA and TS algorithms are presented in section 4.

2. SA HEURISTIC

Local search optimization techniques solve problems by moving from one (current) solution to another (neighboring). The efficiency depends on deriving neighbors based on the previous solution and on the rule of neighbor acceptance after comparing optimality criteria. Usual local search methods have great chance to get trapped in a local minimum, as most of them accept only better neighbors.

SA approach is widely used for the local search in optimization problems due to its ability to overcome local minimums. The name “*simulated annealing*” derives from the analogy with the physical process of annealing: obtaining highly-structured low-energy states of solid in a heat bath through lowering temperature gradually. In scheduling problems, a schedule corresponds to the solid and a penalty function corresponds to its energy. For detailed description of SA, see [1,4,9].

We use the especial ability of SA to accept, with some decreasing probability $e^{-\frac{(f^* - f')}{t}}$, neighboring solutions which do not improve current solution. Here f^* is a value of the objective function for the current solution and f' is the corresponding value for the neighboring solution; t is a control parameter. This allows SA heuristic to overcome local minimums in most cases.

An idea of the proposed approach is to apply SA to a schedule obtained by algorithm 1 for some initial sequence σ of jobs. Preliminary computational experiments show that applying algorithm 1 to a random

sequence or to a sequence obtained by Panwalkar et al. algorithm gives worse results than applying it to a sequence of jobs ordered by their release dates. So we use the latter sequence as an initial sequence σ .

For our scheduling problem, initial value of $t = t_0$ was chosen from the equality

$$e^{\frac{(f_{\min}^* - f'_{\max})}{t_0}} = 0.999, \text{ where } f_{\min}^* \text{ и } f'_{\max} \text{ are the best and the worst values of optimality criteria}$$

obtained by CCS algorithm from the set of 10000 random problem instances. This condition gives a great chance at early stages to accept neighbors which do not improve current solution. After transition (attempt of moving) from current to neighboring solutions, t is reduced: $t := Rt$. In literature, R usually lies between 0.8 and 0.99. After a number of preliminary computational experiments we obtain that $R=0.85$ gives us the best solutions and the best performance among 0.8, 0.85, 0.9, 0.95, 0.99. So we use $R=0.85$ in our heuristic.

Here are the main steps of the proposed approach.

Construct starting sequence σ of jobs by arranging them in non-decreasing order of their release dates. Take solution (s^o, d^o, f^o) as the best solution where f^o is a value of penalty function for the schedule s^o obtained by applying algorithm 1 for σ and d^o is a due date obtained by algorithm 1.

Take the best solution as the current one. Let f^* be a value of the penalty function for the current solution. Repeat the following transition n times: Obtain sequence σ' for neighboring solution by interchanging two jobs randomly in the sequence of current solution. Apply algorithm 1 to σ' to get the neighboring solution with the value f' of penalty function. If the neighboring solution is not worse than the current one, i.e. $f' \leq f^*$, then take it as a current. If the current solution is better than the best one, i.e. $f^* < f^o$, then take current solution as the best. If neighboring solution is worse than the current one, i.e. $f' > f^*$, then accept it as a current with probability $\text{EXP}((f^* - f')/t)$. Reduce control parameter $t := Rt$.

If cycle 2 repeats $10n$ times or if the best solution was not improved during last 10 cycles then stop: the best solution is the result of heuristic. Else repeat cycle 2 again.

There is a difference in such approach from the usual SA method. Algorithm returns to the best solution after n transitions and waits 10 cycles without the best solution improvement until stop. We choose the limit $10n$ times of repeating cycle 2 since preliminary experiments show that further repeating does not lead to considerable improvement of the results. These limitations allow us to reduce run time of the algorithm considerably.

In the worst case it takes $O(n^3)$ units of time to run the algorithm. Detailed description of the algorithm is given below.

ALGORITHM 2.

1. Set $t := t_0$. Construct sequence $\sigma = ([1],[2], \dots, [n])$ by arranging jobs in non-decreasing order of their release dates and determine a schedule $s^*(\sigma)$ and a due date $d^*(\sigma)$ using algorithm 1.
2. Calculate penalty function for the schedule $s^*(\sigma)$ and the due date $d^*(\sigma)$:

$$f^* = f(d^*, s^*).$$
3. Let schedule $s^*(\sigma)$, due date $d^*(\sigma)$ and penalty function value f^* be current solution, and let the best solution $(s^o, d^o, f^o) := \text{current solution } (s^*, d^*, f^*)$.
4. $\text{CyclesWOImpr} := 0$; $\text{Cycles_counter} := 0$.
5. $i := 0$; $\text{Cycles_counter} := \text{Cycles_counter} + 1$.
6. $i := i + 1$.
7. Exchange jobs in position i and randomly chosen position j in the sequence σ and obtain neighboring solution $(s^*(\sigma'), d^*(\sigma'), f')$ for new sequence σ' :

8. Determine $d^*(\sigma')$ and $s^*(\sigma')$ using algorithm 1 for sequence σ' . Calculate $f' = f(d^*(\sigma'), s^*(\sigma'))$.
9. Compare the neighboring solution with the current:
 - 9.1 **if** $f' \leq f^*$ **then** current solution := the neighboring solution.
 - 9.2 **if** $f' < f^o$ **then** the best solution := the neighboring solution; $CyclesWOImpr := 0$.
 - 9.3 **if** $f' > f^*$ **then if** $\text{RANDOM}[1] < \text{EXP}((f^* - f')/t)$ **then** current solution := the neighboring solution.
10. Reduce control parameter $t := Rt$.
11. **if** $i < n$ **then goto step 6**.
12. Current solution := the best solution; $CyclesWOImpr := CyclesWOImpr + 1$;
13. Check stop conditions:
 - 13.1 **if** $CyclesWOImpr = 10$ **then goto step 15**.
 - 13.2 **if** $Cycles_counter = n^2$ **then goto step 15**.
14. **goto step 5**.
15. Current solution is the best solution obtained by heuristic. **stop**.

Notations:

$Cycles_counter$ variable counts number of iteration cycles (steps 6-9) with the same t value.

$CyclesWOImpr$ variable counts number of cycles passed without improving the best (not current) solution.

$\text{RANDOM}[1]$ is the function that returns random real number from the range $[0,1]$.

3. TABU SEARCH

Tabu search (TS) is a local search method which uses special tabu list of visited solutions to prevent searching in the previously explored areas. Together with the best solution, TS keeps information on the itinerary through the last solutions visited [6,7,8]. Unlike the greedy algorithm of local search, in TS the visited but not accepted solutions are placed into the tabu list, and algorithm passes by solutions that are in the tabu list. When the list is full, some solution (the best or the first in tabu list) is removed from the list to accept the last visited solution. The efficiency of such approach depends on a structure of the set of the problem possible solutions. The length of tabu list and its structure affect the speed of problem solving and the quality of the obtained results.

For our problem, the neighbor solution is obtained by exchanging randomly chosen pairs of jobs. After preliminary experiments the length of tabu list is set to $n\sqrt{n}$ and the parameter for defining the size of neighborhood is chosen to be $n/5$. To spread the exploration effort over different region of the set of solutions, we choose parameter \sqrt{n} to get a diversified sequence by exchanging job pairs. The number of the diversifications is chosen to be $n/5$.

In the worst case it takes $O(n^4)$ units of time to run the algorithm. Detailed description of the algorithm is given below

ALGORITHM 3

- 1 Construct sequence $\sigma = ([1],[2],\dots,[n])$ by arranging jobs in non-decreasing order of their release dates. Determine a schedule $s^*(\sigma)$ and a due date $d^*(\sigma)$ using algorithm 1 and calculate $f^*(s^*(\sigma), d^*(\sigma))$. Let the best solution $(s^o, d^o, f^o) := \text{current solution } (s^*, d^*, f^*)$.
- 2 $Restarts_number = 0$.

- 3 $i = 1$.
- 4 $j = 0$. Let the best solution in the neighborhood $(s'', d'', f'') := \text{current solution } (s^*, d^*, f^*)$.
- 5 $j = j + 1$. Exchange job i with randomly chosen job in the sequence of the current solution to obtain new sequence σ' and the corresponding solution $(s(\sigma'), d(\sigma'), f')$ by algorithm 1.
- 6 **if** obtained solution $(s(\sigma'), d(\sigma'), f')$ exists in the Tabu list **then** move it to the end of the list and **then goto step 5**.
- 7 **if** $f' < f''$ **then** the best solution in the neighborhood $(s'', d'', f'') := \text{obtained solution } (s(\sigma'), d(\sigma'), f')$; **else** put obtained solution into the Tabu list.
- 8 **if** list length $> n\sqrt{n}$ **then** remove the first solution from the list.
- 9 **if** $j < n/5$ **then goto step 5**.
- 10 **if** $f'' < f^*$ **then** current solution $(s^*, d^*, f^*) := \text{the best in the neighborhood solution } (s'', d'', f'')$; **else** put solution (s'', d'', f'') into the Tabu list.
- 11 $i = i + 1$; **if** $i \leq n$ **then goto step 4**.
- 12 **if** $f^* < f^o$ **then** the best solution $(s^o, d^o, f^o) := \text{current solution } (s^*, d^*, f^*)$.
- 13 Get new (deversified) sequence by exchanging randomly chosen \sqrt{n} job pairs.
- 14 $Restarts_number = Restarts_number + 1$. **if** $Restarts_number < n/5$ **then goto step 3**.
- 15 The result is the best solution (s^o, d^o, f^o) . **stop**.

Notations:

i, j are the cycles counters.

$Restarts_number$ is the number of diversifications.

4. COMPUTATIONAL EXPERIMENT

Algorithms 1, 2, 3 were programmed in Visual C++ 6.0 and tested on computer with 1800 MHz Athlon XP processor.

Problem instances were created randomly using benchmarks for single machine scheduling problems by Biskup and Feldmann [2]. There are four control parameters for generating problem instances:

n is the number of jobs ($n = 20, 50, 100, 1000$).

p_{\max} is the maximal job processing time ($p_{\max} = 20, 50, 100$).

$r_{\max} = knp_{\max}$ is the maximal release date ($k = 0.01, 0.03, 0.05, 0.1, 0.2, 0.3, 0.5, 1, 1.5, 2, 1/n$).

μ_{\max} is the maximum for α, β, γ values ($\mu_{\max} = 10, 20, 30$).

For each problem instance, α, β, γ are chosen randomly from the range $[1, \mu_{\max}]$ taking into account $\beta > \gamma$ condition; p_j and r_j are chosen from the ranges $[1, p_{\max}]$ and $[1, r_{\max}]$, respectively. Preliminary experiments show that the run time of the algorithms depend on r_{\max} but not on p_{\max} and μ_{\max} . For problem instances with $n = 1000$, smaller values of p_{\max} were taken to avoid memory overflow : $p_{\max} = 6, 16, 33$.

To compare SA, TS and CCS heuristics we calculate f_{sa} / f_{ccs} , f_{ts} / f_{ccs} and f_{ts} / f_{sa} , where f_{sa} , f_{ts} and f_{ccs} are the penalty function values obtained by the SA, TS and CCS algorithms for the same problem instances.

It is natural that the proposed heuristics obtain better solution than those by CCS algorithm for the same problem instances. For all combinations of n and k , we have $f_{sa}/f_{ccs} < 1$ and $f_{ts}/f_{ccs} < 1$ for all problem instances. The range of average ratios f_{sa}/f_{ccs} is from 0.535 to 0.966; the f_{ts}/f_{ccs} ratios lay in the range from 0.495 to 0.966. Note that for small k , results obtained with SA and TS heuristics are closer to those obtained with CCS heuristic. This can be explained by the nature of CCS algorithm: using a sequence for the relaxed problem with zero ready times, it obtains solutions close to optimal when release dates r_j are distributed in a small range.

The average value of f_{ts}/f_{sa} is close to 1 (Table 1), though for small n values SA approach gives slightly better solutions than those by TS heuristic.

Tables 2 and 3 show the run time of SA and TS algorithms for $n = 20, 50, 100, 1000$ depending on release dates (parameter k). Average amount of time necessary for solving the same problem instance is smaller for TS algorithm.

For large values of n ($n = 1000$) TS algorithm obtains better solutions though it takes much more run time to solve a problem comparing to SA heuristic.

5. SUMMARY

We consider single machine common due date assignment and scheduling problem under given release dates for the jobs. The problem is known to be *NP*-hard. An $O(n \log n)$ heuristic algorithm was proposed for the problem by Cheg, Chen and Shakhlevich [3] (CCS algorithm). We compare simulated annealing (SA) and tabu search (TS) approaches to the problem. Preliminary computational experiments led to choosing a sequence of jobs ordered by their release dates as an initial solution in both approaches. In deriving neighbors we also took into account release dates of the jobs. The number of iterations was restricted to get $O(n^3)$ running time for SA algorithm and $O(n^4)$ time for TS algorithm.

Computational experiments show that both approaches can be used for the problem under consideration giving better results than CCS algorithm, and for smaller number n of jobs ($n \leq 100$) SA algorithm gives better solutions yielding to TS algorithm in time. For larger number of jobs the results are opposite: TS algorithm leads to better solutions taking more time. Further, a comparison of the performance with other meta-heuristic approach based on genetic algorithm can be considered.

ACKNOWLEDGEMENTS

The research was partially supported by INTAS project 03-51-5501.

REFERENCES

1. Aarts, E.H.L., Korst, J.H.M., van Laarhoven P.J.M.: Simulated Annealing. In: Aarts, E.H.L., Lesnra, J.K. (Eds.), *Local Search in Combinatorial Optimization*. Wiley, Chichester, 1997, 91-120.
2. Biskup, D., Feldmann, M.: Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers & Operations Research* 28 (2001), 787-801.
3. Cheng, T.C.E., Chen, Z-L., Shakhlevich, N.V.: Common due date assignment and scheduling with ready times. *Computers & Operations Research* 29 (2002), 1957-1967.
4. Dowsland, K.A. Simulated Annealing. In: Reeves C.R. (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*. Blackwall, Oxford (1993), 20-69.

5. Feldmann, M., Biskup, D.: Single machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers & Industrial Engineering* 44 (2003), 307-323.
6. Glover, F.: Tabu search: part I. *ORSA Journal of computing* 1 (1989), 190-206.
7. Glover, F.: Tabu search: part II. *ORSA Journal of computing* 2 (1990), 4-32.
8. Hertz, A., Taillard, E., de Werra, D.: Tabu Search. In: Aarts, E.H.L., Lenstra, J.K. (Eds.), *Local Search in Combinatorial Optimization*. Wiley, Chichester, 1997, 121-136.
9. Kirkpatrick, C.D. Gelatt, Jr., Vecchi, M.P.: Optimization by simulated annealing. *Science* 220 (1983), 671-680.
10. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and scheduling: Algorithms and complexity. In: Graves, S.C., Zipkin, P.H., Rinnooy Kan, A.H.G. (Eds.), *Logistics of Production and Inventory*. Handbooks in Operations Research and Management Science, vol. 4, North-Holland, Amsterdam, 1993, 445– 522.
11. Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P.: Complexity of machine scheduling problems. *Annals of Discrete Mathematics* 1 (1977), 343–62.
12. Panwalkar, S.S., Smith, M.L., Seidmann, A.: Common due date assignment to minimize total penalty for the one machine scheduling problem. *Operations Research* 30 (1982), 391–399.

TABLES

Table 1. f_{is} / f_{sa} value for each n depending on k .

| K | $n=20$ | $n=50$ | $n=100$ | $n=1000$ |
|-------|--------|--------|---------|----------|
| $1/n$ | 1.018 | 1.003 | 1.001 | 1 |
| 0.01 | 1.021 | 1.002 | 1.001 | 1 |
| 0.03 | 1.016 | 1.003 | 1.001 | 0.998 |
| 0.05 | 1.016 | 1.002 | 1 | 0.996 |
| 0.1 | 1.013 | 1.003 | 1 | 0.992 |
| 0.15 | 1.015 | 1.005 | 0.999 | 0.982 |
| 0.2 | 1.014 | 1.009 | 1 | 0.985 |
| 0.25 | 1.029 | 1.011 | 1.001 | 1.104 |
| 0.3 | 1.024 | 1.016 | 1.005 | 0.997 |
| 0.5 | 1.042 | 1.022 | 1.017 | 0.925 |
| 0.7 | 1.013 | 1.012 | 1.007 | 1 |
| 1.0 | 1.010 | 1.005 | 1.003 | 0.986 |
| 1.5 | 1.002 | 1.003 | 1 | 0.962 |
| 2.0 | 1.017 | 1.001 | 1 | 0.77 |

Table 2. Run time (in seconds) of SA algorithm to solve the problem for each n depending on k .

| K | $n=20$ | $n=50$ | $n=100$ | s |
|-------|--------|--------|---------|-----|
| $1/n$ | 0.001 | 0.014 | 0.151 | 260 |
| 0.01 | 0.001 | 0.014 | 0.152 | 259 |
| 0.03 | 0.001 | 0.013 | 0.151 | 258 |
| 0.05 | 0.001 | 0.012 | 0.148 | 260 |
| 0.1 | 0.001 | 0.011 | 0.140 | 259 |
| 0.15 | 0.001 | 0.009 | 0.128 | 259 |
| 0.2 | 0.001 | 0.008 | 0.094 | 260 |
| 0.25 | 0.001 | 0.007 | 0.068 | 259 |
| 0.3 | 0.001 | 0.007 | 0.056 | 258 |
| 0.5 | 0.001 | 0.005 | 0.031 | 113 |
| 0.7 | 0.001 | 0.004 | 0.024 | 151 |
| 1.0 | 0.001 | 0.004 | 0.027 | 174 |
| 1.5 | 0.001 | 0.004 | 0.029 | 194 |
| 2.0 | 0.001 | 0.004 | 0.034 | 190 |

Table 3. Run time (in seconds) of TS algorithm to solve the problem for each n depending on k .

| K | $n=20$ | $n=50$ | $n=100$ | $n=1000$ |
|-------|--------|--------|---------|----------|
| $1/n$ | 0 | 0.003 | 0.059 | 2292 |
| 0.01 | 0 | 0.003 | 0.061 | 2846 |
| 0.03 | 0 | 0.003 | 0.042 | 3646 |
| 0.05 | 0 | 0.003 | 0.040 | 3900 |
| 0.1 | 0 | 0.002 | 0.028 | 4661 |
| 0.15 | 0 | 0.002 | 0.027 | 5709 |
| 0.2 | 0 | 0.002 | 0.020 | 11837 |
| 0.25 | 0 | 0.002 | 0.018 | 6968 |
| 0.3 | 0 | 0.002 | 0.017 | 6834 |
| 0.5 | 0 | 0.001 | 0.009 | 5467 |
| 0.7 | 0 | 0.001 | 0.007 | 3887 |
| 1.0 | 0 | 0.001 | 0.007 | 3304 |
| 1.5 | 0 | 0.001 | 0.009 | 3324 |
| 2.0 | 0 | 0.001 | 0.009 | 2146 |