Implementation of an efficient parametric algorithm for optimal scheduling on parallel machines with release dates

To cite this article: Y A Mezentsev et al 2019 J. Phys.: Conf. Ser. 1333 022002

View the article online for updates and enhancements.
Implementation of an efficient parametric algorithm for optimal scheduling on parallel machines with release dates

Y A Mezentsev¹, I V Estraykh¹, N Y Chubko¹

¹ Novosibirsk State Technical University, Prospekt Karla Marks 20, Novosibirsk, Russia

E-mail: ive7@yandex.ru

Abstract. An original statement and solution algorithms are presented for one of the key problems in the scheduling theory. The problem of optimal scheduling for a parallel system consists in the generation or control of schedules to minimize the schedule length or the losses from schedule disruptions in the completion of jobs on machines. This problem is NP-hard and cannot be solved exactly for any real-life number of dimensions. A series of modifications of the efficient parametric algorithm is proposed to find an approximate solution, which are an extension of a similar algorithm for optimal scheduling on unrelated parallel machines with release dates using the performance criterion (Stakh). Software implementations of the algorithm modifications have been tested on the data of a generating problem by the Stakh criterion; the corresponding statistics is provided.

1. Introduction

The scheduling problem under consideration for a one-stage system of unrelated parallel machines, as well as its applications, is relatively well known [1–4, 6].

Efficient algorithms have been proposed for solving a series of modifications of the problem [2–4]. However, the development of algorithms with acceptable accuracy and speed remains a pertinent unresolved issue for many of the problem modifications.

Specifically, one of the traditional applications of these algorithms is airline operations planning. In this case, planning involves a series of stages, the key ones being: aircraft scheduling, fleet assignment, routing, and crew planning. A detailed review on this topic exists, e.g., in [1]; formal problem statements and main approaches to solving the key problems are discussed in [2] and in numerous publications following individual lines of applied research from among those listed above [2–5].

2. Problem Statement for Optimal Scheduling on Parallel Machines with Release Dates

We consider a series of jobs that should be distributed between parallel machines with known (different) completion times to minimize the total time of completion of all the jobs (minimize the completion time for the entire system; or minimize the total costs; or maximize the job quality, measured by some indicator). Preemptions are forbidden.

We also assume that the release dates in the system of parallel machines are known at the initial time. In this case, when assigning jobs to machines, one should take into account the release dates. We designate the release date for job i on machine j as \( r_{ij}^0 \) and arrange the jobs for each machine in the increasing order of \( r_{ij}^0 \). Then, \( T^\theta = \left\lVert r_{ij}^\theta \right\rVert_j, \quad j = 1, \ldots, J, \quad i = 1, \ldots, I \) (I is the number of jobs at the input of the
system; \( J \) is the number of machines) can be interpreted as a schedule at the input of each machine in the system. Here and below, the \( \| \) symbol stands for a vector, matrix, or tensor, depending on the dimension context.

Then, \( t_{i,j} \) is the fixed time of completion of job \( i \) on machine \( j \), \( T = \| t_{i,j} \|, j = \overline{1,J}, i = \overline{1,I} \). Moreover, \( b_j \) and \( \overline{b}_j \) are, respectively, the minimum and maximum number of jobs assigned to machine \( j \).

Applying the above notation, the formal statement for the problem of optimal scheduling on parallel machines with release dates is written as follows [7]:

\[
\sum_{j=1}^{J} x_{i,j} = 1, \quad i = \overline{1,I} \tag{1}
\]

\[
b_j \leq \sum_{i=1}^{I} x_{i,j} \leq \overline{b}_j, \quad j = \overline{1,J} \tag{2}
\]

\[
x_{i,j} = 1, \text{ if job } i \text{ is assigned to machine } j,
\]

\[
x_{i,j} = 0 \text{ otherwise},
\]

\[
y_{i,j} \geq 0, \quad i = \overline{1,I}, \quad j = \overline{1,J},
\]

\[
\tau_{i,j} = \tau_{i,j}^a - \sum_{k=1}^{K} (t_{i,j} + t_{k,j}) x_{i,j}, \quad i = \overline{1,I}, \quad j = \overline{1,J},
\]

\[
\bar{\tau}_{i,j} = \tau_{i,j} + y_{i,j} \geq 0, \quad i = \overline{1,I}, \quad j = \overline{1,J},
\]

\[
\sum_{i=1}^{I} \bar{\tau}_{i,j} x_{i,j} + \sum_{i=1}^{I} t_{i,j} x_{i,j} \leq \lambda, \quad j = \overline{1,J},
\]

where \( x_{i,j} \) are variables to be determined, representing the assignment of job \( i \) to machine \( j \). The values \( \bar{\tau}_{i,j} \) are the actual delays in the completion of job \( j \) by machine \( i \) after the completion of the previous job assigned to the machine. The compensating variables \( y_{i,j} \) are introduced to avoid negative release dates \( \tau_{i,j} \). Indeed, the presence of inequalities (4) and (6) guarantees the fulfillment of the conditions:

\[
\bar{\tau}_{i,j} = \begin{cases} \tau_{i,j}, & \text{if } \tau_{i,j} \geq 0, \\ 0 & \text{otherwise.} \end{cases}
\]

Conditions (1)–(3) are typical of the job assignment problem. Condition (1) ensures that any job \( i \) is assigned to one machine only. Condition (2) ensures that no less than \( b_j \) and no more than \( \overline{b}_j \) jobs are assigned to any machine \( j \). Conditions (5)–(6) calculate the actual delays in the completion of job \( i \) by machine \( j \) after the completion of the previous job assigned to this machine.

If a schedule quality (or efficiency) criterion, e.g.

\[
\lambda \rightarrow \min
\]

exists for constraints (1)–(7), then relations (1)–(8) define a problem of synthesizing performance-optimal schedules for a system of parallel machines with release dates. Relations (7) and (8) represent the minimax criterion, meaning the maximum performance criterion (sometimes called balanced load criterion), of the form \( \max \{\sum_{i=1}^{I} (t_{i,j} + \bar{\tau}_{i,j}) x_{i,j}\} \rightarrow \min \).

The schedule synthesized by solving problem (1)–(8) is fully determined by the optimal assignments \( x^*_{i,j}, i = \overline{1,I}, j = \overline{1,J} \) and the actual delays for these assignments \( \bar{\tau}_{i,j}^*, i = \overline{1,I}, j = \overline{1,J} \).

We should point out the main difficulties associated with solving problem (1)–(8). These are, primarily, Boolean and continuous variables, recursions in (5) and (7), and knapsack constraints. Taken together, these features put the problem into the class of NP-hard problems of mixed integer programming.
3. One-Parameter Algorithm to Search for Suboptimal Solutions

Owing to the recursions, DP may well be the only method directly applicable to problem (1)–(8). However, this method is inefficient when applied “as is.” An attempt to find an exact solution of (1)–(8) by DP leads to a complete enumeration of all admissible options. One can easily count the total number $N$ of these options. For example, let $k$ be the step number and $b_j = 0$ and $\overline{b}_j = 1$, $j = 1, J$ in (2). Then, as shown below, considering that the number of options grows in a geometric progression with DP steps, we have: $N = \left( J^{I+1} - J \right)/2$. Therefore, when applied to problem (1)–(8), DP has greater-than-exponential complexity and cannot be applied “as is” to problems with any real-life number of dimensions.

To construct an efficient approximate algorithm, we use the general DP scheme with the sifting of the locally worst options at certain DP steps. This approach has been tested by the authors on problems of optimal scheduling on unrelated machines with release dates [2] to show good practical results in terms of accuracy and speed.

We assume that all the jobs $i = 1, I$ for each machine are arranged in the order of the initial release dates (input schedule $\left\| e^0_i \right\|$). Then, we use the DP procedure to find the step numbers: $\eta = 1, I$. We denote the time when machine $j$ completes job $\eta$ at step $\eta$ as $f_{\eta,j}(\tilde{e}_{\eta,j}, t_{\eta,j}, x_{\eta,j})$, $j = 1, J$, and the conditionally minimal time of completion of all the jobs at steps from 1 to $\eta$ as $\varphi_{\eta}(\tilde{e}_{\eta,j}, t_{\eta,j}, x_{\eta,j})$, $j = 1, J$, $i = 1, \eta$.

\[ f_{\eta,j}(\tilde{e}_{\eta,j}, t_{\eta,j}, x_{\eta,j}) = \max \left\{ \left[ x_{\eta,j} - \varphi_{\eta-1,j}(\tilde{e}_{\eta-1,j}, t_{\eta,j}, x_{\eta,j}) \right] + \left[ t_{\eta,j} - x_{\eta,j} \right] \right\}, \ \eta = 1, I, \ \eta = 1, J, \ \eta = 1, I. \quad (9) \]

The recurrent Bellman relation for this problem is:

\[ \varphi_{\eta,j}(\tilde{e}_{\eta-1,j}, t_{\eta,j}, x_{\eta,j}) = \left\{ \left[ f_{\eta,j}(\tilde{e}_{\eta,j}, t_{\eta,j}, x_{\eta,j}) + \varphi_{\eta-1,j}(\tilde{e}_{\eta-1,j}, t_{\eta,j}, x_{\eta,j}) \right] \right\}, i = 1, \eta, \quad (10) \]

\[ \varphi_{\eta}(\tilde{e}_{\eta,j}, t_{\eta,j}, x_{\eta,j}) = \max \left\{ \left[ \varphi_{\eta,j}(\tilde{e}_{\eta,j}, t_{\eta,j}, x_{\eta,j}) \right] \right\}, j = 1, J, \ \eta = 1, I. \quad (11) \]

To achieve the maximum performance by criterion (7)–(8), one should choose at the last step the minimum value of $\varphi_{1,j}(\tilde{e}_{1,j}, t_{1,j}, x_{1,j})$, i.e., find $\lambda = \min \left\{ \varphi_{1,j}(\tilde{e}_{1,j}, t_{1,j}, x_{1,j}) \right\}, j = 1, J, \ \eta = 1, I$. We now calculate the total number of schedule options that we should find to guarantee the best schedule:

\[ N = J + J^2 + \ldots + J^k + \ldots + J^I = \left( J^{I+1} - J \right)/2. \quad (12) \]

The total number of intermediate schedules $N'$ is:

\[ N' = J + J^2 + \ldots + J^{k-1} + J^I + \ldots + J^I = \left( J^I - J \right)/2 + (I - k) \cdot J^I. \quad (13) \]

Since $k$ is a constant, relation (13) represents the polynomial dependence of the complexity of the parametric DP algorithm with option-sifting on the dimension of problem (1)–(8).

4. Three-Parameter Algorithm

These modifications introduce two additional parameters when eliminating the locally worst options. These parameters control the sizes of the assignment sets in general and individual steps of the DP procedure in particular. We now determine the parameters of the algorithm.

Let $T_j = \frac{1}{I} \sum_{i=1}^{I} t_{i,j}$ be the average completion time of job $j$ by all machines. We specify the following parameters:
\( \gamma = 1, 2, ..., \) : this parameter specifies the depth of counting at each step, defined as the number of jobs assigned at one step of the algorithm (e.g., if \( \gamma = 3 \), then at step one, we consider all assignment options for jobs 1, 2, 3; at step two, for jobs 4, 5, 6, etc.).

\( K \) : this parameter was defined above (\( J^{k+1} \geq K > J^{k} \), where \( k \) is the parameters of the \( A_{R_1} \) algorithm); it sets the number of jobs with the locally best system completion time at the current step, which are chosen for the next step.

\( \delta \geq 1 \) : this parameter sets the height of the barrier to cut the locally inefficient options, leaving only those for assignment of which the machine completion time does not exceed \( \delta \cdot T^{av}_j \). As a result, those jobs are selected that satisfy the conditions \( t_{i,j} \leq \delta \cdot T^{av}_j \), \( j = 1, J, i = 1, I \).

**Algorithm \( A_{R_3} \) (three-parameter)**

1. Enter the input data \( (r^0_{i,j}, t_{i,j}) \), \( j = 1, J, i = 1, I \), and the parameters \( \gamma \), \( \delta \), \( K \). Assuming that \( \phi_{0,j}(r^0_{i,j}, t_{i,j}, x_{i,j}) = 0 \), set the initial step number \( \eta = 0 \).
2. Increase the step number \( \eta := \eta + \gamma \).
3. Check the step number. If \( \eta \geq I - \gamma \), then go to point 8; otherwise, go to the next point.
4. Assign jobs from \( \eta - \gamma + 1 \) to \( \eta \) to machines \( (j = 1, J) \) in all possible combinations, considering the recurrent relations (9)–(11) and the constraints \( t_{i,j} \leq \delta \cdot T^{av}_j \), \( j = 1, J, i = 1, I \).
5. For all the obtained assignment options, calculate \( f_{\eta,j}(\tilde{r}_{\eta,j}, t_{\eta,j}, x_{\eta,j}) \) and schedule lengths \( \phi_{\eta,j}(\tilde{r}_{\eta,j}, t_{i,j}, x_{i,j}) \).
6. Sort the assignment options in the ascending order of the schedule lengths \( \phi_{\eta,j}(\tilde{r}_{\eta,j}, t_{i,j}, x_{i,j}) \).
7. Select the first \( K \) assignment options, sifting out the remaining ones. Return to point 2.
8. Assign the remaining \( I - \gamma \) jobs to machines in all possible combinations, considering the recurrent relations (9)–(11) and the constraints \( t_{i,j} \leq \delta \cdot T^{av}_j \), \( j = 1, J, i = 1, I \).
9. Choose schedule options with the minimum length. Construct the final schedules using the inverse DP procedure.

**5. Improving Procedures**

Based on the computational experiments with software implementations of the \( A_{R_1}, A_{C_3}, \) and \( A_{R_3} \) algorithms, we developed a series of additional heuristic procedures that can improve solutions of problem (1)–(8) instances.

Specifically, the results of \( A_{R_3} \) can be improved using several values of \( \gamma \). Since the computational experiments for different \( \gamma \) are independent of one another, we can use all the advantages of modern multicore processors by computations of each \( \gamma \) as an independent flow. Since the computing time increases with increasing \( \gamma \) (as with increasing \( K \)), other parameters being equal, we can use a dynamic increase in the \( K \) parameter for small \( \gamma \). By varying the \( \gamma \) and \( K \) parameters, we can equalize the end time for all the flows, simultaneously improving the accuracy of the \( A_{R_3} \) algorithm. We use the following notation: \( K_p \) is the value of the \( K \) parameter for flow \( p \); \( \gamma_p \) is the value of the \( \gamma \) parameter for flow \( p \), \( p = 1, P \), where \( P \) is the number of flows.
Since the $A_{P_3}$ algorithm does not guarantee optimal solutions either, we can apply a modification of the exchange-based improving algorithm $A_C$, taking account the specific features of $A_{P_3}$. We denote this algorithm as $A_{C,p}$.

**Algorithm $A_{C,p}$**

1. Choose the best solutions obtained by the $A_{P_3}$ algorithm in the multiflow version for each flow $p$ ($x_{i,j}^p = 1_{I,P}$).
2. Set the step number $\eta = 0$ and the parameter for the depth of job exchange between the machines $rl$ in the range $1 \leq rl \leq I$.
3. Increase the step number $\eta := \eta + 1$.
4. Check the stopping criterion by the step number. If $\eta \geq I$, then go to point 6. Otherwise, take the next point.
5. The jobs from $(\eta - 1) \cdot rl + 1$ to $\eta \cdot rl$ (or to $I$ if $\eta \cdot rl > I$) are distributed between the machines by the exchange algorithm $A_C$ for all $p = 1,P$. We choose the locally best option in terms of schedule length $\phi_{\eta,j}(\tilde{r}_{\eta,j},\tilde{t}_{i,j},x_{i,j})$. All other jobs from $\eta \cdot rl + 1$ to $I$ are not reassigned. Return to point 3.
6. Save the last assignment and the schedule length estimate $x_{i,j}^*, \lambda = \phi_{\eta,j}(\tilde{r}_{\eta,j},\tilde{t}_{i,j},x_{i,j}^*)$. End of the algorithm.

6. Test Results for Software Implementations of the Algorithms

The tables below (Tables 1, 2) show the test results for software implementations of the above algorithms.

The designations of the algorithm parameters in the tables are the same as those in the text above.

We use 2 groups of test examples produced by random number generators (10 tests in each group). The test dimensions by group are as follows: 5 machines, 100 jobs in the first group; 10 machines, 100 jobs in the second group.

**Table 1.** Test results for $A_{P_3} + A_{C,p}$ (test dimension 5x100) $rl = 6$, $\gamma_p = \frac{24}{2}$.

| No. | $\delta = 1$ | $\delta = 1.2$ | Record | Improvement |
|-----|--------------|----------------|--------|-------------|
| $K$ | 50 | 150 | 200 | 250 | 50 | 150 | 200 | 250 | 350 | 350 | 350 | 350 | 350 | 1 |
| 1 | 351 | 351 | 350 | 350 | 356 | 351 | 351 | 351 | 351 | 350 | 350 | 1 |
| 2 | 328 | 330 | 327 | 330 | 329 | 330 | 328 | 330 | 330 | 330 | 330 | 3 |
| 3 | 367 | 366 | 369 | 366 | 372 | 365 | 364 | 364 | 364 | 5 |
| 4 | 356 | 356 | 353 | 353 | 353 | 356 | 356 | 356 | 356 | 349 | 349 | 2 |
| 5 | 332 | 328 | 331 | 329 | 330 | 330 | 330 | 330 | 330 | 328 | 328 | 3 |
| 6 | 353 | 349 | 351 | 354 | 358 | 349 | 359 | 359 | 355 | 349 | 14 |
| 7 | 393 | 394 | 393 | 394 | 397 | 394 | 394 | 394 | 394 | 390 | 390 | 3 |
| 8 | 380 | 379 | 379 | 378 | 380 | 379 | 380 | 379 | 379 | 378 | 378 | 1 |
| 9 | 369 | 367 | 367 | 367 | 367 | 367 | 367 | 366 | 366 | 366 | 1 |
| 10 | 378 | 378 | 378 | 378 | 385 | 377 | 377 | 377 | 377 | 377 | 0 |

Time: 0:57 1:48 2:15 2:39 1:05 2:13 2:49 3:23
Table 2. Test results $A_{P_3} + A_{C_p}$ (dimension $10 \times 100$) $rl = 6$, $\gamma_p = 2,4$.

| No. | $\delta = 1$ | $\delta = 1.2$ | Record | Improvement |
|-----|--------------|----------------|--------|-------------|
| $K$ | 50           | 150            | 200    | 250         | 50 | 150 | 200 | 250 |
| 1   | 155          | 145            | 144    | 145         | 144 | 144 | 3   |
| 2   | 118          | 118            | 118    | 123         | 118 | 122 | 121 | 118 | 0   |
| 3   | 138          | 142            | 139    | 140         | 140 | 141 | 141 | 138 | 3   |
| 4   | 130          | 127            | 126    | 127         | 126 | 126 | 127 | 126 | 0   |
| 5   | 143          | 142            | 141    | 141         | 141 | 139 | 139 | 139 | 2   |
| 6   | 134          | 132            | 129    | 129         | 134 | 129 | 131 | 130 | 129 | 3   |
| 7   | 144          | 139            | 139    | 138         | 120 | 139 | 138 | 138 | 0   |
| 8   | 132          | 132            | 129    | 135         | 136 | 134 | 130 | 130 | 129 | 0   |
| 9   | 139          | 136            | 136    | 132         | 137 | 135 | 131 | 135 | 131 | 0   |
| 10  | 122          | 120            | 120    | 121         | 119 | 120 | 120 | 120 | 119 | 0   |

Time 0:54 1:53 2:21 2:45 1:04 2:14 2:45 3:31

Tables 1, 2 show the solution estimates (values of the objective functions) $\lambda^*$. The columns marked as No. show the number of the test in the group. The Time columns show the operation time of the corresponding algorithm modification, measured in minutes and seconds (in the min:s format). The dimensions for the test group in Table 1 are as follows: 5 machines, 100 jobs ($5 \times 100$); in Tables 2: 10 machines, 100 jobs ($10 \times 100$). Tables 1 and 2 show the results obtained by applying the algorithms $A_{P_3} + A_{C_p}$ (with different combinations of the parameters).

The Improvement columns show the relative improvement in the solution results achieved by applying $A_{P_3} + A_{C_p}$ (with the use of the $A_{C_p}$ improving procedures) in comparison with $A_{P_3}$.

Evidently, in most cases the above described modifications of the parametric and improving algorithms demonstrate considerable advantages over the previously developed algorithms. A slight worsening in the last two tests in Table 1, compared with the results of $A_{P_3}$ and $A_{P_3} + A_{C_p}$, respectively, does not disprove the general trend.

Another conclusion, which lies on the surface, is that the stricter the resource constraints of the problem (here, the job–machine ratio), the greater the advantage of the developed tools near the optimum of the solutions.

The computing time for all the tests turned out to be acceptable and, in fact, linearly dependent on the test dimension, which experimentally confirms the efficiency of the algorithms.

These results indirectly indicate the existence of conditions that control the complexity and accuracy of the applied algorithms for optimal scheduling on unrelated parallel machines with constraints on the time of decision-making. Specifically, the accuracy of solutions can be improved by controlling the resource constraints rather than increasing the complexity of the algorithm.

7. Conclusions

The main results of this work are the development, software implementation, and comparative analysis of modifications of algorithms to search for approximations to optimal solutions of the NP-hard problem of scheduling on unrelated parallel machines with release dates by the Stakh criterion.

Analysis of the test results for software implementations of the algorithms for tests with a close-to-realistic number of dimensions allows us to conclude that the developed tools are directly applicable in real-life contexts. These include the control over aircraft assignments and flight schedules for airline companies [1, 5], target assignments for missile defense systems, drilling of oil and gas condensate fields [3], and many other problems [6].
8. Acknowledgments
This work was supported by the Ministry of Science and Higher Education of the Russian Federation, project no. 2.2327.2017/4.6, and by Novosibirsk State Technical University, project no. TP-EI-1_17.

References
[1] Grönkvist M 2005 *The Tail Assignment Problem*. (Göteborg: Chalmers University of Technology and Göteborg University)
[2] Avdeenko T V, Mesentsev Y A 2016 *IFAC-PapersOnLine* 49(12) 1743-1748
[3] Avdeenko T V, Mezentsev Y A, Estraykh I V 2017 *IFAC-PapersOnLine* 50(1) 13096-13101
[4] Lenstra J K, Shmoys D B, Tardos E 1987 *Approximation algorithms for scheduling unrelated parallel machines* (Amsterdam: Centre for Mathematics and Computer Science)
[5] Rosenberger J M 2002 *Topics in Airline Operations* (Atlanta: School of Industrial & Systems Engineering, Georgia Institute of Technology)
[6] Vazirani V 2001 *Approximation Algorithms* (Springer) p 375
[7] Santos F C, Vilarinho P M In *Proc. WCE 2010* 3 (London, U.K.) 2443–48