Efficient Method for Single Machine Total Tardiness Problem

Sergii Minukhin
Kharkiv National University of Economics, Kharkiv, Ukraine
ms_vl@mail.ru

***Abstract*— Method for solving the problem of minimizing the total tardiness of jobs with due dates is considered as a problem of determining shortest Hamiltonian path in the graph. Computational results for random generated instances with up to 150 jobs for unweighted and weighted total tardiness problem show the good performance and the efficiency of the developed method and algorithms.**

***Keywords*— total tardiness; due dates; NP-hard; graph; Hamiltonian path; dominance rule; error**

I. INTRODUCTION

The formulation of the problem for composing an optimal sequence of jobs (schedule) stands as follows. A set of independent jobs $J = \{j_1, j_2, \ldots, j_n\}$ entered into the single machine (processing element) and every job will be executed on it without interruption. For each job its processing time l_j and due date d_j are given. The problem is to determine the order (sequence) of execution of all submitted jobs that enter a computing element at the same time, which will be minimize their total tardiness time.

II. PROBLEM FORMULATION

Mathematically, the problem of minimization of total tardiness for the jobs with due dates is formalized as follows: it is needed to build a jobs execution schedule that minimizes the objective function

$$f = \sum_{j=1}^{n} \max(0, C_j - d_j) \to \min, \qquad (1)$$

where C_j denotes the real completion time of the job j, d_j denotes the job due date, n denotes number of jobs. Comparative analysis of methods and algorithms for solving the problem of minimizing the total time tardiness is in Table 1.

The review of methods for solving this problem is given in [1 – 7]. According to [1, 3, 6, 7], this problem is NP-hard. In article [2] algorithms of dynamic programming are reviewed for solving the problem. However, these methods have a general disadvantage: the solving of the problem in real-time is complicated because of the exponential growth of alternatives scheduling that being analyzed.

TABLE I. COMPARATIVE ANALYSIS OF METHODS

Method	Complexity	Author
Decomposition/dynamic programming	$O(n^4 \sum p_i)$	Lawler [1]
Decomposition/dynamic programming	50 jobs	Baker and Schrage [2]
Decomposition/dynamic programming	$O(n^7/\varepsilon)$	Lawler [3]
Dynamic programming	with up100 jobs	Potts and Van Wassenhove [4]
Bound improvement	$O(n^6 \log n + n^6/\varepsilon)$	Kovalyov [5]
Odd-even partition	$O(n \sum p_i)$ for special case	Lazarev [7]
Rounding procedure and bound improvement	$O(n^5 \log n + n^5/\varepsilon)$	Koulamas [6]

III. PROBLEM FORMALIZATION ON GRAPH BASED REPREZENTATION

To formalize the model and the method of solving the assigned problem let us consider the graph G, where each vertex corresponds to the job, and the edge (i, j) with the weight characteristics $T_j = \max(0, C_j - D_j)$ (Fig. 1).

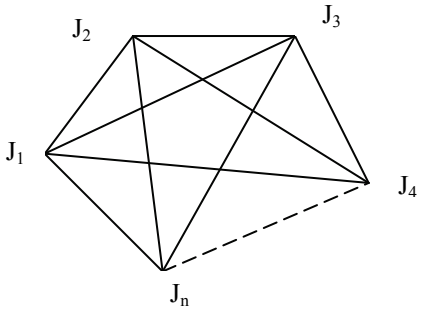

Figure 1. Graph G

Then the problem of minimization can be considered as a problem of determination of the traversal of all vertices of G with minimal length of path. Thus, this statement leads to the problem of determining the shortest Hamiltonian path in the graph G [8].

The distinction of this problem is the fact that the weights of graph edges, arising from an arbitrary vertex j_i, are not constant, but depend on the order of vertices traversal, leading to the j_i, as it affects the value C_j that is determined by the sum completion time jobs that precede the execution of jobs j_i, i.e.:

$$C_j = \sum_{l_i \in \mu_{ij}} l_i .$$

Let all the possible states of some system be defined by graph G(V, E) with n vertices, where vertices correspond to possible states of the system. Let us go to the space of $n \cdot * (n-1)$ states. To do this, we associate with each n state other $(n-1)$ states, that characterize the way to reaching the next state, which consists of a set {1, 2,..,n}. Tree of all paths of graph G contains n horizontal lines and $(n-1)$ layers. During the determination of paths, the passage of each horizontal line allowed only one time in accordance with the task of finding the shortest Hamiltonian path. Thus, using the graph G and having rules for the formation paths of the next rank, paths of the arbitrary ranks can be built step-by-step from arbitrary vertex s up to rank $r = n-1$. With this solving approach, the problem of determining the shortest Hamiltonian path in the graph G it is necessary to construct the shortest path with rank $r = n-1$ from vertex 1 ,2,…,n to the rest of the vertices of a graph and then select the shortest among them.

Let the graph G consists of 4 vertices, which represent the 4 works with the following characteristics (Table 2). Weight of vertex J of the graph is determined by $\max(0, C_j - D_j)$. In the first layer of a spanning tree drawn in the graph paths will be arranged by weighs of J_1, J_2, J_3, J_4; on the second – the paths of rank 2: S_{J1J2}, S_{J1J3}, S_{J1J4}, S_{J2J1}, S_{J2J3}, S_{J2J4}, S_{J3J1}, S_{J3J2}, S_{J3J4}, S_{J4J1}, S_{J4J2}, S_{J4J3}, connecting the 2 vertices, etc.; in the fourth layer are all paths of a graph of rank 4, which uses a combination of all $S_{JiJk,}$ i.e., the length of all paths of the rank 4 connecting the 4 vertices. From these all paths of the rank 4 of a graph select the path of minimal length.

TABLE II. EXAMPLE FOR 4 VERTEX GRAPH G

	J_1	J_2	J_3	J_4
Lj	52	48	58	24
dj	1140	439	0	754
Cj	729	777	835	859
Tj	0	338	835	105

IV. COMPUTATIONAL TESTING AND ANALYSIS OF RESULTS

For testing the application in order to increase sufficiency of gained results the library [9] and random sequence generator were used. The developed generator produces sequences with different characteristics.

The instances are generated as follows: for each job j ($j = \overline{1, N}$), an integer processing time l_j was generated from the uniform distribution [1, 100]. In order to simulate different hardness the due dates are generated from different uniform distributions. For an average tardiness factor TF = {0.2, 0.4 ,0.6 0.8, 1.0} and a relative range of due dates RDD = {0.2, 0.4 ,0.6 0.8, 1.0}, from the uniform distribution

$$\left[L \cdot \left(1 - TF - \frac{RDD}{2}\right), L \cdot \left(1 - TF + \frac{RDD}{2}\right)\right] \qquad (2)$$

an integer due date d_j is generated for each job, where

$$L = \sum_{j=1}^{N} l_j . \qquad (3)$$

Thus, for each observation 25 instances are generated.

To determine the sufficient number of instances in a sample, 1, 2, 3 and 4 samples with 25, 50, 75 and 100 instances respectively were generated with 40, 60, 80, 100, 120, 150 jobs. The schedules for the samples were built with direction optimization algorithm, whereupon mean and standard deviation for minimum, maximum and average tardiness time were calculated.

Results of research justify practical implication: 50 and 75 instances in a sample are enough for it to be sufficient because mean and standard deviation for 75 instances (in some cases for 50 instances) don't vary significantly comparing to 100 instances.

In order to estimate the accuracy of heuristic algorithms approximation comparing to complete search one, mean of relative error was calculated as:

$$M_\varepsilon = \frac{\sum_{j=1}^{N} \frac{t_j^{opt} - t_j^{comp}}{t_j^{comp}}}{N}. \qquad (4)$$

where t_j^{opt} denotes tardiness time of the job obtained from direction optimization algorithm, t_j^{comp} denotes tardiness time of the job obtained from complete search algorithm.

To estimate time complexity of reviewed algorithms (complete search, direction optimization, and direction optimization with dominance (dispatching) rule computing experiments were performed.

In the case when choosing the next vertex for the Hamiltonian path some vertices (jobs) have the same length, then choose the vertex to which the dominance (dispatching) rule. The dispatching rule is EDD – the next job scheduled is a job that has the earliest due date among the jobs. The sequence is the same as the sequence of jobs arranged in the ascending order of their due dates.

In Table 2 results of the experiment with number of jobs from 3 to 11 are presented. The schedules were built with complete search, direction optimization, and direction optimization with dominance rule algorithms.

TABLE III. RESULTS OF COMPUTATIONAL EXPERIMENTS

Number of jobs	Mean of relative tardiness time error for optimization algorithm, %	Mean of relative tardiness time error for optimization algorithm with dominance rule, %
3	26,69	26,69
4	3,35	2,35
5	11,13	11,79
6	21,51	35
7	25,65	25,5
8	10,11	8,21
9	12,51	10,93
10	27,66	36,77
11	3,73	5,23

The approximation functions for execution times were built using cubic polynomials for complete search algorithm (5), for direction optimization algorithm (6), and for direction optimization algorithm with dominance rule EDD (7):

$$T(N)_{comp.search} = 51820.05n^3 - 944767.87n^2 + 5329583.96n - 9176552.53,$$

$$T(N)_{opt.} = 0.03n^3 - 0.22n^2 + 1.18n - 2.02, \quad (6)$$

$$T(N)_{opt.D} = 0.05n^3 - 0.55n^2 + 3.22n - 5.72. \quad (7)$$

Thus, performed experiments proved obtained in [8] theoretical time complexity of reviewed algorithms $O(n^3)$. On Fig. 2, 3 execution time dependencies on number of jobs for direction optimization algorithm (algorithm 1) and direction optimization algorithm with dominance rule (algorithm 2) are presented.

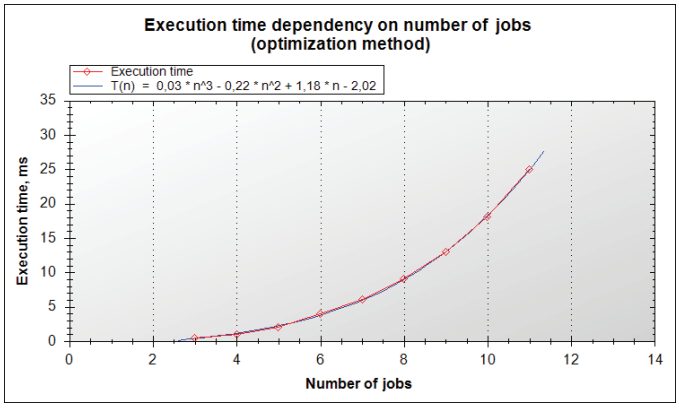

Figure. 2. Execution time dependency for algorithm 1

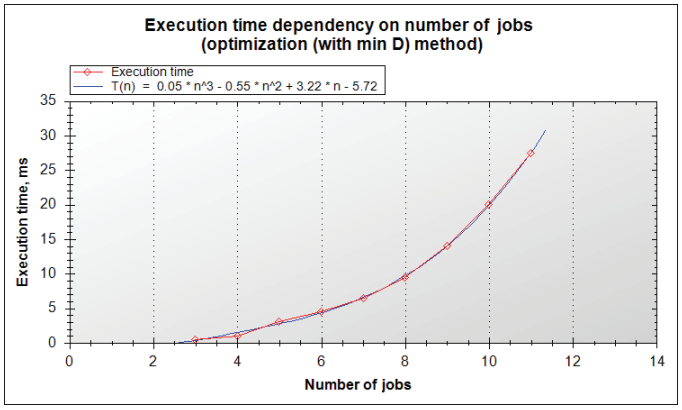

Figure. 3. Execution time dependency for algorithm 2

On Fig. 4, 5 relative errors comparing to complete search algorithm for two proposed algorithms – heuristic algorithm and heuristic algorithm with dominance rule respectively are presented; on Fig. 6 dependency time execution on number of jobs for heuristic algorithm 1 and algorithm 2 are presented.

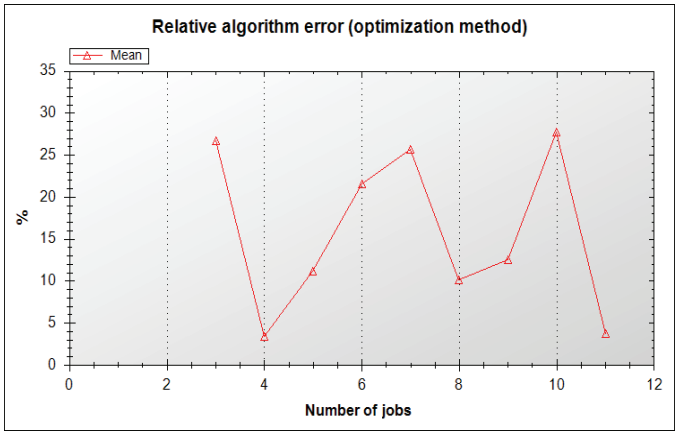

Figure. 4. Plot of relative error of heuristic algorithm 1 comparing to complete search algorithm

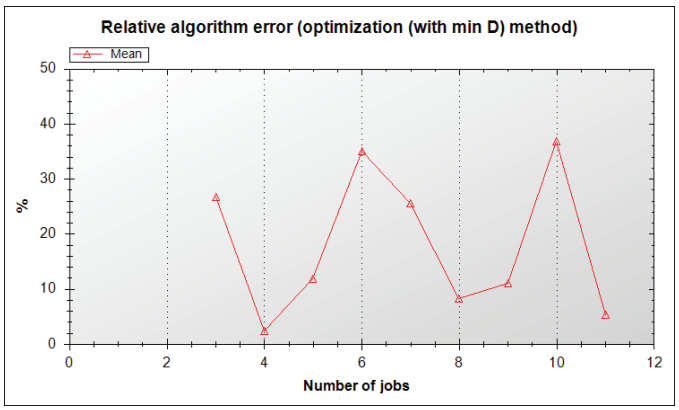

Figure. 5. Plot of relative error of heuristic algorithm 2 comparing to complete search algorithm

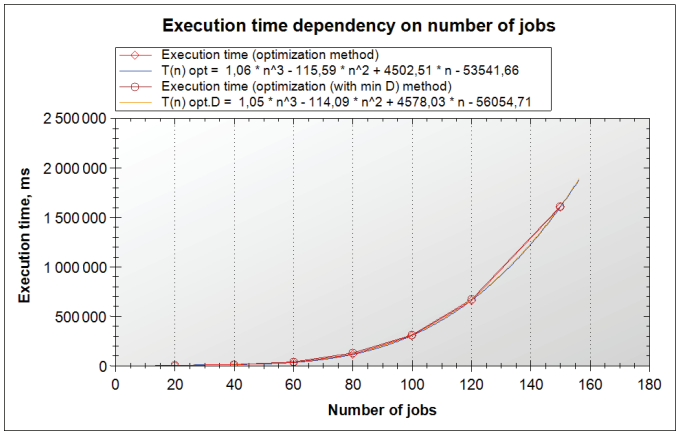

Figure. 6. Plot of dependency time execution on number of jobs for heuristic algorithm 1 and algorithm 2

The single machine total weighted tardiness problem is defined as follows. Consider n jobs to be processed on a single machine that can handle only one job at a time. Each job j, available for processing at time zero, has a positive processing time p_j, a positive weight w_j, and a positive due date d_j. For a given sequence of jobs, the tardiness of job j is defined as

$T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of job j. The objective of the total weighted tardiness problem is to find a processing order of all the jobs; this order is a schedule that minimizes the sum of the weighted tardiness of all jobs:

$$\sum_{j=1}^{n} w_j T_j,$$

where w_j generated by the random uniform in the interval [1, 10].

For arbitrary positive weights, it is strongly NP-hard [10].

In the case when choosing the next vertex for the Hamiltonian path some vertices (jobs) have the same length, then choose the vertex to which the dominance (dispatching) rule.

The dispatching rules are:

WSPT: calculate ratio $S_j = p_j / w_j$. Jobs are scheduled in the ascending order of the ratios;

BWF (Biggest Weight First). Jobs are scheduled in the descending order of their weights.

Plot of relative error of heuristic algorithm comparing to

complete search algorithm for total weighted tardiness is shown on Fig. 7.

A plot of execution time on the amount of jobs for the weighted case is shown in Fig. 8.

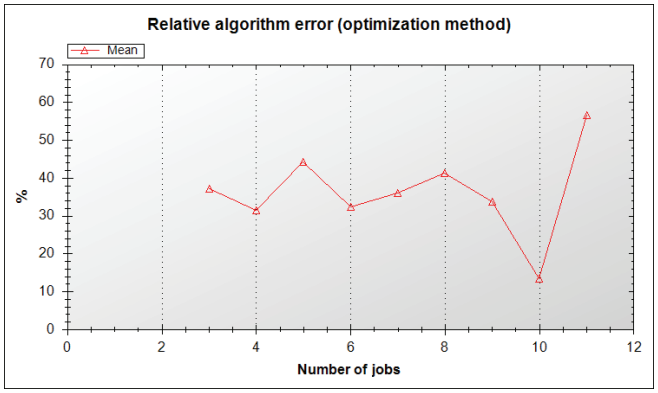

Figure. 7. Plot of relative error of heuristic algorithm comparing to complete search algorithm for total weighted tardiness

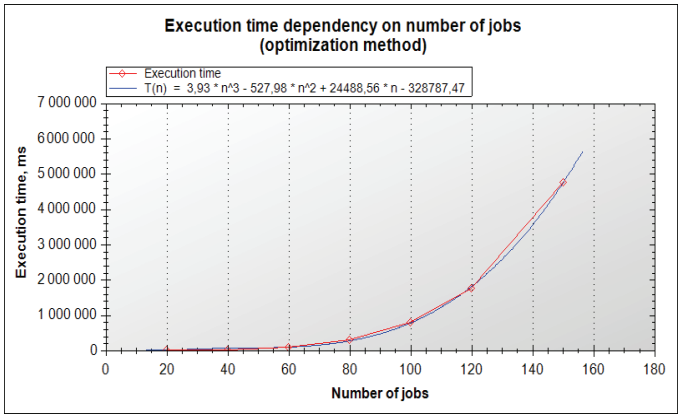

Figure. 8. Plot of dependency time execution on number of jobs for heuristic algorithm for total weighted tardiness

As can be seen from Fig. 8, time complexity for the weighted case as determined by the value $O(n^3)$ and allows you to extend the results to any number of works entering the single machine.

The experiment was conducted on a personal computer with dual-core processor with a frequency of 3.2 GHz and 4 GB of RAM under OS Windows 7. To ensure the accuracy of time measurement experiments were computed on a single processor core.

V. CONCLUSIONS

1. Theoretically derived time complexity $O(n^3)$ of solving the problem minimization of total tardiness on a single machine on rank based method on direction optimization algorithms [8] has been experimentally substantiated.

2. The program for generating random testing samples has been developed. It allows conducting computing experiment and a qualitative analysis of test results for different representative test samples.

3. Program implementation of the complete search algorithm and rank based method two direction optimization algorithms allows evaluating the results and analyzing the relative error of the developed methods.

The time complexity of the method, resulting from a computer experiments, justifies the possibility application of the proposed algorithms for real-time computing systems.

REFERENCES

[1] E.L. Lawler. A ''pseudo polynomial'' algorithm for sequencing jobs to minimize total tardiness. // Annals of Discrete Mathematics, No 1, 1977, pp. 331–342.

[2] K.R. Baker, L.E. Shrage. Finding an optimal sequence by dynamic programming an extension to precedence-related tasks. // Oper. Res, No 26, 1978, pp. 111–120.

[3] E.L. Lawler. A fully polynomial approximation scheme for the total tardiness problem. // Operations Research Letters, No 1, 1982, pp. 207–208.

[4] C.N Potts, L.N. Van Wassenhove. Dynamic programming and decomposition approaches for the single machine total tardiness problem. // European Journal of Operational, No 32, 1987, pp. 405–414.

[5] M.Y. Kovalyov Improving the complexities of approximation algorithms for optimization problems. // Operations Research Letters, No 17, 1995, pp. 85–87.

[6] C. Koulamas A faster fully polynomial approximation scheme for the single machine total tardiness problem. // European Journal of Operational Research, No 193, 2009, pp. 637–638.

[7] A.A. Lazarev, E.R. Gafarov. Special case of the single machine total tardiness problem is NP-hard. // In: 12th IFAC Symposium on Information Control Problems in Manufacturing. INCOM'2006.Preprints, Vol. III, Operational Research, May 17–19, 2006, Saint-Etienne, France, pp. 155–157.

[8] S.V. Minukhin The method of minimizing of the compute time of the tasks with due dates on no clustered resource of computing system. // Information management systems on railway transport, No 3, 2009, pp. 47 – 53.

[9] OR-Library: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html.

[10] N. Liu, M. Abdelrahman, S. Ramaswamy. A Genetic Algorithm for Single Machine Total Weighted Tardiness Scheduling Problem // International journal of intelligent control and systems, vol. 10, No. 3, september 2005, pp. 218–225.