UDC 621.398

S. Minukhin
Kharkiv National University of Economics,

Department of Information Systems

ALGORITHMS OF MINIMIZING THE TOTAL TARDINESS ON A SINGLE MACHINE BASED ON DETERMINATION OF SHORTEST HAMILTONIAN PATH IN THE GRAPH AND THE DOMINANCE RULES

The paper proposes a method, algorithms and its implementations using dominance rules for minimizing the total tardiness on a single machine based on shortest Hamiltonian path in a arbitrary graph that improve the efficiency and not reduce the execution time. Metrics for evaluating the effectiveness of the dominance rules are proposed. The experimental results of algorithms are developed that justify the effectiveness of the proposed modifications by getting local optimal solutions during procedure.

Keywords - Hamiltonian path, graph, the optimal schedule, total tardiness, due date, weight, dominance rule

Introduction

Modern information and communication systems are defined by the presence of a great number of different types of resources – informational, computing, etc., its distribution, the control of which requires increase of efficiency and improvement of control over distributed processing systems, particularly on local resources. One of the goals is to develop efficient algorithms for scheduling and optimization of performance for the selected criteria in real-time computing machines (e.g., Grid system nodes, workstations (Network of workstations, desktop Grids), etc.). Solving the problem of optimization of local resource performance of such systems in order to reduce, and possibly even avoid the tardiness in cases where the task is characterized by due date. This will reduce the risks associated with the ability to pay fines, which lead to economic losses of users. The reviews of solving the problem of minimizing the total tardiness are examined in [1–4], the basic methods of its solution are given in [5–11]. A method of minimizing the total tardiness (method of direction optimization) based on the construction of the shortest Hamiltonian path in the fully-connected graph and based on the ranked approach with time complexity of O(n3) is shown in article [12]; in articles [13, 14] it is hypothesized that in cases where there are several alternative paths, the certain path is to be selected so where the vertex of graph that is added to the current rank of the Hamiltonian path has the shorter due date.

The goal of this paper – is to develop and analyze the algorithms for constructing an optimal schedule tasks to minimize the total tardiness on a single machine (computing device) based on the construction of the shortest Hamiltonian path in an arbitrary graph and dominance rules along with metrics of evaluation the effectiveness of their use.

Formulation of the problem

Assume that the single machine receives a set of independent jobs $J = \{J_1, J_2, ..., J_n\}$, and each of them is being continuously executed on it. The duration of each job L_j and its due date d_j are known as well. The problem is to determine the order (sequence) of execution of all submitted jobs that enter a machine (device) at the same time, which will minimize the total tardiness of the input queue of all tasks $TT_S = \sum_{\{j \in S\}} \max(0, C_j - d_j)$, where C_j denotes the completion time of the job j. This problem is defined by $1||\sum T_i$, for solution of which the time complexity is obtained in [12, 13] and the dominance rules are first suggested for obtaining local optimum, namely the EDD rule (Earliest Due Date). In order to substantiate the hypothesis and to prove it experimentally, there are some considerable examples that

illustrate the essence of the method and the use of dominance rules, on the basis of a fully-connected graph with vertices that characterize the operation of test sequence.

Suppose that there is a set of jobs J_1 (4, 7), J_2 (5 , 6), J_3 (3 , 7), J_4 (3 , 4), for which it is necessary to build an optimal schedule. Each job J is characterized by two parameters: L – competition time, d – due date, and thus it appears as J (L, d).

The graph that reflects the sequence of jobs is shown in Fig. 1.

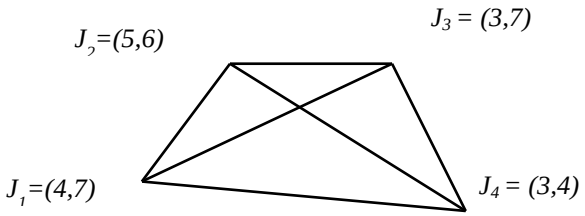

Fig. 1. Full-connected graph with 4 vertices (jobs)

Algorithms to minimize the total tardiness along with dominance rules

For an explanation of the algorithms operation a matrix is used, in which the columns correspond to the ranks (layers), and rows correspond to lines, the number of which is determined by the amount of jobs for which the schedule (timetable) is built (see Table 1). To characterize the value of the current tardiness of the set of jobs on each rank the notation $T(j)_{j1j2\ldots jn}$ is used, where j – is the number of rank (column), $j_1j_2,\ldots,j_n$ – is number of jobs (operations) that are the part of the current path to any job on current rank (column) of matrix. Thus, the total tardiness of all jobs will be determined on the last rank.

At the beginning of the procedure jobs are ordered arbitrarily, the tardiness of each of them is equal to 0, and thus, their total tardiness is equal to 0. On the first rank (in the first row) of matrix all the paths from the jobs of J_2, J_3, J_4 to job J_1 are constructed, in second row the paths from the jobs J_1, J_3, J_4 to job J_2 are constructed, in the third row the paths from the jobs J_1, J_2, J_4 to job J_3 is constructed, in the fourth row the paths from the jobs J_1, J_2, J_3 to job J_4 is constructed. Then according to the algorithm it is necessary to choose the minimum paths for each row: for row 1 – $T(2)_{3,1}$ =3+(4-7) = 0 and $T(2)_{4,1}$ = 3+(4-7) = 0; for row 2 – $T(2)_{3,2}$ = 3+(5-6) = 2 and $T(2)_{4,2}$ = 3+(5-6) = 2; for row 3 – $T(2)_{4,3}$ = 3+(3-7) = -1; for row 4 – $T(2)_{3,4}$ = 3+(3-4) = 2.

After that, on the rank 2 the following is chosen: in the first row – $T(3)_{4,3,1}$ = 6+(4-7) = 3 and $T(3)_{3,4,1}$ = 6+(4-7) = 3; in the second row – $T(3)_{4,3,2}$ = 6+(5-6) = 5 and $T(3)_{3,4,2}$ = 6+(5-6) = 5; in the third row – $T(3)_{4,1,3}$ = 7+(3-7) = 3; in the fourth row – $T(3)_{3,1,4}$ = 7+(3-4) = 6. Then, a path from the vertices of the rank 3 to the vertices of the rank 4 is built: for the row 1 – $TT_{4,3,2,1}$ = 0+0+5+8 = 13; for the row 2 – $TT_{4,3,1,2}$ = 0+0+3+9 = 12 and $TT_{4,1,3,2}$ = 0+0+3+9 = 12. So, the optimal Hamiltonian path (see Figure.1) include the sequence of jobs $J_4J_3J_1J_2$. Thus, from the Table.1 it follows that there are «competing» paths from which can choose only one – accidentally or by using the dominance rules (for example EDD [1–3, 11, 13]), which give the local optimal solution, or by «stretching» all the paths to the next rank, which means selecting all the constructed paths (without cutting off) for the next rank. If the current values of tardiness are equal to each other on any rank, a dominance rule EDD is used: for example, in rank 3 $T(3)_{4,3,1}$ = 6+(4-7) = 3 and $T(3)_{3,4,1}$ = 6+(4-7) = 3. The choice falls on $T(3)_{4,3,1}$ because $d_4 < d_3$. The procedure continues until it reaches the rank number 4, where the shortest Hamiltonian path with the minimum tardiness is finally selected. This will be the total tardiness, i.e. the desired solution. So for this example the optimal schedule is determined by the sequence of jobs J_4, J_3, J_1, J_2 with a total tardiness $TT_{j4j3j1j2}$ = 12. The sequence of construction of Hamiltonian path in the graph in Fig. 1 at each rank is shown in Fig. 2–4 (Hamiltonian path is bold in the graph). The example of using the proposed algorithm with dominance rule EDD is shown in the Table 1.

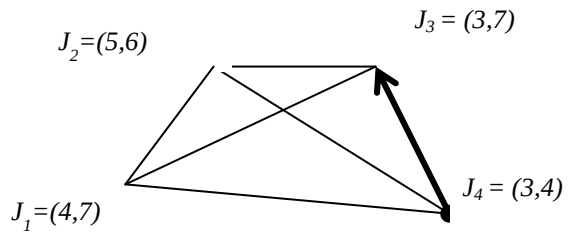

Fig. 2. Choosing the vertex at the rank 2

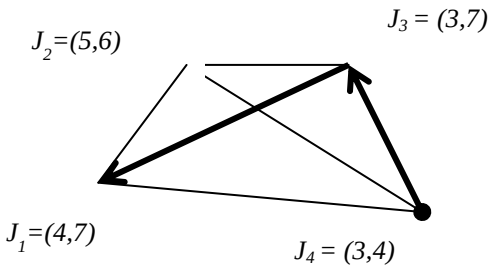

Fig. 3. Choosing the vertex at the rank 3

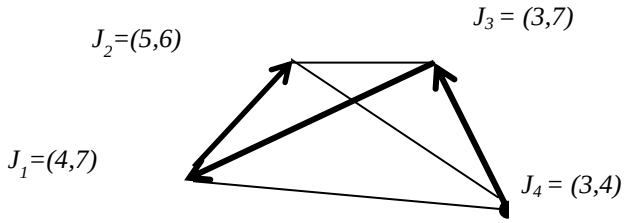

Fig. 4. Choosing the vertex at the rank 4

An optimal paths in a graph in all over matrices in Tables 1–3 and Tables 5, 6 are shown in bold.

An example of the algorithm using the criterion of MDD rule $\min\{\max(p_j, d_j - t)\}$ [15] is shown in the Table 2, the example of using the proposed algorithm without dominance rules is shown in the Table 3.

Table 1.

An example of algorithm operation including EDD rule

Vertex	Rank 1	Rank 2	Rank 3	Rank 4
J_1 (4;7)	$T(1)_1=(0+4)-7=-3$	$T(2)_{21}=(5+4)-7=2$ **$T(2)_{31}=(3+4)-7=0$** **$T(2)_{41}=(3+4)-7=0$**	$T(3)_{421}=(8+4)-7=5$ **$T(3)_{431}=(6+4)-7=3$** **$T(3)_{341}=(6+4)-7=3$**	$TT_{3421}=0+2+5+8=15$
J_2 (5;6)	$T(1)_2=(0+5)-6=-1$	$T(2)_{12}=(4+5)-6=3$ **$T(2)_{32}=(3+5)-6=2$** **$T(2)_{42}=(3+5)-6=2$**	$T(3)_{412}=(7+5)-6=6$ **$T(3)_{432}=(6+5)-6=5$** **$T(3)_{342}=(6+5)-6=5$**	**$TT_{4312}=0+0+3+9=12$** **$TT_{4132}=0+0+3+9=12$**
J_3 (3;7)	$T(1)_3=(0+3)-7=-4$	$T(2)_{13}=(4+3)-7=0$ $T(2)_{23}=(5+3)-7=1$ **$T(2)_{43}=(3+3)-7=-1$**	**$T(3)_{413}=(7+3)-7=3$** $T(3)_{423}=(8+3)-7=4$	
J_4 (3;4)	$T(1)_4=(0+3)-10=-7$	$T(2)_{14}=(4+3)-4=3$ $T(2)_{24}=(5+3)-4=4$ **$T(2)_{34}=(3+3)-4=2$**		

Table 2.

An example of algorithm operation including MDD rule

Vertex	Rank 1	Rank 2	Rank 3	Rank 4
J_1 (4;7)	$T(1)_1=(0+4)-7=-3$	$T(2)_{21}=(5+4)-7=2$ **$T(2)_{31}=(3+4)-7=0$** **$T(2)_{41}=(3+4)-7=0$**	$T(3)_{421}=(8+4)-7=5$ **$T(3)_{431}=(6+4)-7=3$** **$T(3)_{341}=(6+4)-7=3$**	$TT_{4321}=0+0+5+8=13$
J_2 (5;6)	$T(1)_2=(0+5)-6=-1$	$T(2)_{12}=(4+5)-6=3$	$T(3)_{412}=(7+5)-6=6$	**$TT_{4312}=0+0+3+9=12$**

		$T(2)_{32}=(3+5)-6=2$ **$\underline{T(2)_{42}=(3+5)-6=2}$**	**$\underline{T(3)_{432}=(6+5)-6=5}$** **$T(3)_{342}=(6+5)-6=5$**	$TT_{4132}=0+0+3+9=12$
J_3 (3;7)	$T(1)_3=(0+3)-7=-4$	$T(2)_{13}=(4+3)-7=0$ $T(2)_{23}=(5+3)-7=1$ **$\underline{T(2)_{43}=(3+3)-7=-1}$**	**$\underline{T(3)_{413}=(7+3)-7=3}$** $T(3)_{423}=(8+3)-7=4$	
J_4 (3;4)	$T(1)_4=(0+3)-4=-1$	$T(2)_{14}=(4+3)-4=3$ $T(2)_{24}=(5+3)-4=4$ **$\underline{T(2)_{34}=(3+3)-4=2}$**		

Table 3.

An example of algorithm operation without dominance rules

Vertex	Rank 1	Rank 2	Rank 3	Rank 4
J_1 (4;7)	$T(1)_1=(0+4)-7=-3$	$T(2)_{21}=(5+4)-7=2$ **$\underline{T(2)_{31}=(3+4)-7=0}$** $T(2)_{41}=(3+4)-7=0$	$T(3)_{321}=(8+4)-7=5$ **$T(3)_{431}=(6+4)-7=3$** **$\underline{T(3)_{341}=(6+4)-7=3}$**	**$TT_{4321}=0+0+5+8=13$**
J_2 (5;6)	$T(1)_2=(0+5)-6=-1$	$T(2)_{12}=(4+5)-6=3$ **$\underline{T(2)_{32}=(3+5)-6=2}$** $T(2)_{42}=(3+5)-6=2$	$T(3)_{312}=(7+5)-6=6$ **$\underline{T(3)_{432}=(6+5)-6=5}$** $T(3)_{342}=(6+5)-6=5$	$TT_{3412}=0+0+3+9=14$ $TT_{3142}=0+0+6+9=15$
J_3 (3;7)	$T(1)_3=(0+3)-7=-4$	$T(2)_{13}=(4+3)-7=0$ $T(2)_{23}=(5+3)-7=1$ **$\underline{T(2)_{43}=(3+3)-7=-1}$**		
J_4 (3;4)	$T(1)_4=(0+3)-4=-1$	$T(2)_{14}=(4+3)-4=3$ $T(2)_{24}=(5+3)-4=4$ **$\underline{T(2)_{34}=(3+3)-4=2}$**	**$\underline{T(3)_{314}=(7+3)-4=6}$** $T(3)_{324}=(8+3)-4=7$	

Analysis of the results (Tables 1–3) shows that that it is possible to improve the result (total tardiness), and in some cases quite significantly – MDD rule can reduce the value of the total tardiness when using EDD rule and without using up to 30%, which may significantly affect the outcome in case of a large number of jobs in sequence, with the number of vertices of the Hamiltonian path, derived from the rule, is 2 out of the 4 vertices of the full-connected graph (Fig. 1).

The problem of minimizing the total weighted tardiness (*TWT*) on a single machine (device) $TWT_S = \sum_{\{j \in S\}} \max\left(0, \left(C_j - d_j\right) \cdot w_j\right)$ $(1||\sum w_i T_i)$ can be stated as the following. The set of jobs indexed from 1 to n, must be processed without interruption on one computing device that can handle only one job at a time. All jobs come to a device at time that equals to 0. Job is characterized by its processing time L_i (processor time, p_i), tardiness d_i and weight w_i. For convenience, all jobs are ordered according to EDD rule, so that $d_i < d_j$; if $d_i = d_j$, then $p_i < p_j$; if $p_i = p_j$, then $w_i < w_j$ for all i, j $(i < j)$. If job i is completed after its due date d_i, there is a penalty for tardiness (weighted). For experimental research that evaluates the impact of the inclusion of dominance rules, the rules that are listed in the Table 4 are used.

Table 4.

Terms of dominance rules for the total weighted tardiness

Rule	Name	Description	Formula of determining the priority
WEDD	Weighted Earliest Due Date	Weighted with the earliest due date	$\max\left\{\frac{w_j}{d_j}\right\}$
WMDD	Weighted Modified Due Date	Weighted with the modified due date	$\min\left\{\frac{\max\left(p_j, d_j - t\right)}{w_j}\right\}$
WSPT	Weighted Shortest Processing Time	Weighted with shortest processing runtime	$\min\left\{\frac{p_j}{w_j}\right\}$

The examples of algorithms with the inclusion of dominance rule WMDD to algorithm of the weighted case, but not including the rule are given in Table 5 and Table 6, respectively.

Table 5.

An example of algorithm operation with the inclusion of WMDD rule

Vertex	Rank 1	Rank 2	Rank 3	Rank 4
J_1 (4;7,2)	$WI(1)_1 = ((0+4)-7)*2 = -6$	$WI(2)_{21} = ((5+4)-7)*2 = 4$ **$WI(2)_{31} = ((3+4)-7)*2 = 0$** **$WI(2)_{41} = ((3+4)-7)*2 = 0$**	$WI(3)_{321} = ((8+4)-7)*2 = 10$ **$WI(3)_{431} = ((6+4)-7)*2 = 6$** **$WI(3)_{341} = ((6+4)-7)*2 = 6$**	**$TWT_{4321} = 0+0+5\cdot4+8\cdot2 = 36$**
J_2 (5;6,4)	$WI(1)_2 = ((0+5)-6)*4 = -4$	$WI(2)_{12} = ((4+5)-6)*4 = 12$ **$WI(2)_{32} = ((3+5)-6)*4 = 8$** **$WI(2)_{42} = ((3+5)-6)*4 = 8$**	$WI(3)_{312} = ((7+5)-6)*4 = 24$ **$WI(3)_{432} = ((6+5)-6)*4 = 20$** **$WI(3)_{342} = ((6+5)-6)*4 = 20$**	$TWT_{4312} = 0+0+3\cdot2+9\cdot4 = 42$ $TWT_{3142} = 0+0+3\cdot5+9\cdot4 = 51$
J_3 (3;7,5)	$WI(1)_3 = ((0+3)-7)*5 = -20$	$WI(2)_{13} = ((4+3)-7)*5 = 0$ $WI(2)_{23} = ((5+3)-7)*5 = 5$ **$WI(2)_{43} = ((3+3)-7)*5 = -5$**		
J_4 (3;4,3)	$WI(1)_4 = ((0+3)-4)*3 = -3$	$WI(2)_{14} = ((4+3)-4)*3 = 9$ $WI(2)_{24} = ((5+3)-4)*3 = 12$ **$WI(2)_{34} = ((3+3)-4)*3 = 6$**	**$WI(3)_{314} = ((7+3)-4)*3 = 18$** $WI(3)_{423} = ((8+3)-7)*5 = 20$	

Experimental investigation of algorithms and analysis of the results

To perform the experimental research of the proposed algorithms their program realization is developed, through which the dependences of runtime and the total tardiness are calculated for the basic algorithm (optimization direction [12]) and for the algorithms using dominance rules for the unweighted and weighted cases (see Figures 5–8) based on methods used in [14].

Table 6.

An example of algorithm operation without WMDD rule

Vertex	Rank 1	Rank 2	Rank 3	Rank 4
J_1 (4;7,2)	$WI(1)_1 = ((0+4)-7)*2 = -6$	$WI(2)_{21} = ((5+4)-7)*2 = 4$ **$WI(2)_{31} = ((3+4)-7)*2 = 0$** **$WI(2)_{41} = ((3+4)-7)*2 = 0$**	$WI(3)_{421} = ((8+4)-7)*2 = 10$ **$WI(3)_{431} = ((6+4)-7)*2 = 6$** **$WI(3)_{341} = ((6+4)-7)*2 = 6$**	**$TWT_{4321} = 0+6+20+8\cdot2 = 42$**
J_2 (5;6,4)	$WI(1)_2 = ((0+5)-6)*4 = -4$	$WI(2)_{12} = ((4+5)-6)*4 = 12$ **$WI(2)_{32} = ((3+5)-6)*4 = 8$** **$WI(2)_{42} = ((3+5)-6)*4 = 8$**	$WI(3)_{412} = ((7+5)-6)*4 = 24$ **$WI(3)_{432} = ((6+5)-6)*4 = 20$** **$WI(3)_{342} = ((6+5)-6)*4 = 20$**	$TWT_{3412} = 0+6+6+9\cdot4 = 48$ $TWT_{4132} = 0+0+15+9\cdot4 = 51$
J_3 (3;7,5)	$WI(1)_3 = ((0+3)-7)*5 = -20$	$WI(2)_{13} = ((4+3)-7)*5 = 0$ $WI(2)_{23} = ((5+3)-7)*5 = 5$ **$WI(2)_{43} = ((3+3)-7)*5 = -5$**	**$WI(2)_{413} = ((7+3)-7)*5 = 15$** $WI(2)_{423} = ((8+3)-7)*5 = 20$	
J_4 (3;4,3)	$WI(1)_4 = ((0+3)-4)*3 = -3$	$WI(2)_{14} = ((4+3)-4)*3 = 9$ $WI(2)_{24} = ((5+3)-4)*3 = 12$ **$WI(2)_{34} = ((3+3)-4)*3 = 6$**		

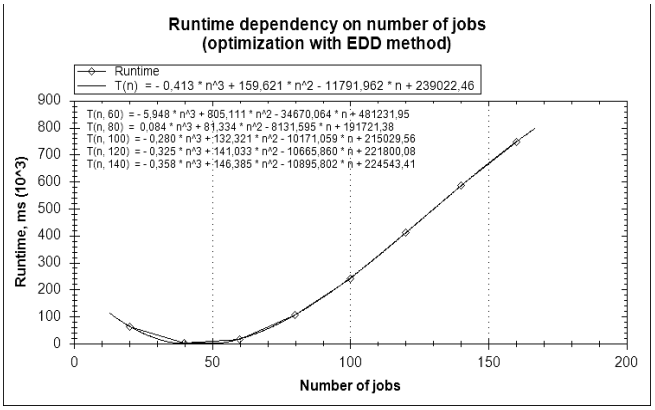

Fig. 5. The runtime dependency on number of jobs for the basic algorithm using dominance rules EDD and MDD

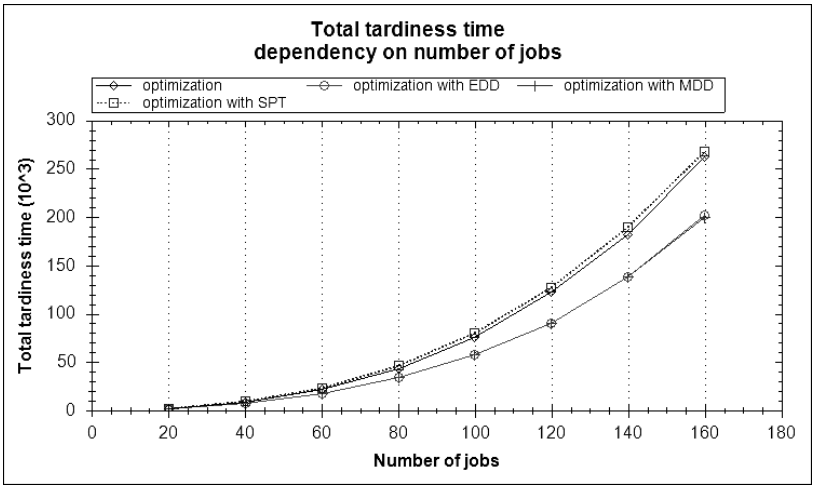

Fig. 6. The total tardiness dependency on number of jobs for the basic and enumeration algorithms using dominance rules EDD and MDD

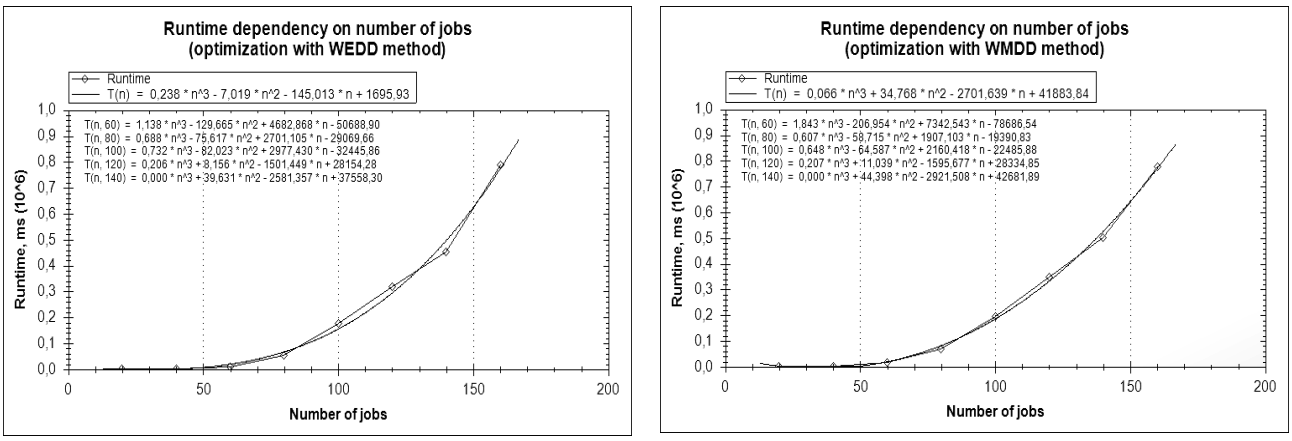

Fig. 7. The runtime dependency on number of jobs using dominance rules WEDD and WMDD

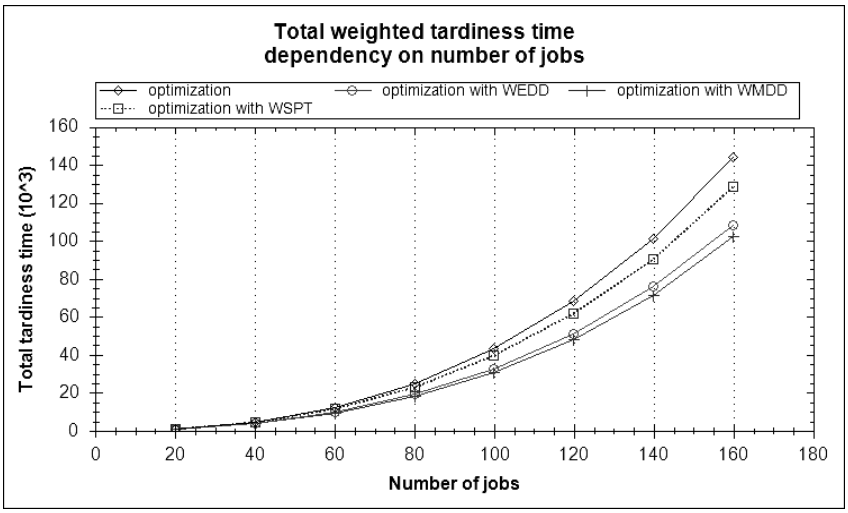

Fig. 8. The total weighted tardiness dependency on number of jobs for the basic and the complete enumeration algorithms using dominance rules WEDD and WMDD

Evaluating the effectiveness of the inclusion of dominance rules to the basic algorithm

To evaluate of the effectiveness of the inclusion the dominance rules to the basic algorithm for obtaining local optimum (in optimization using a certain dominance rule) is defined two metrics: the ratio of the number of vertices of the Hamiltonian path, derived from the rules of dominance, to the total number of vertices – density $\sum_{GP} v_{dr} \Big/ n$, where v_{dr} – is a set of vertices on the graph included in Hamilton path, obtained on the basis of the dominance rule; the relative reduction of tardiness $(TT - TT_{dr}) / TT$, where TT_{dr} – is total tardiness with the inclusion of dominance rules to the algorithm; TT – total tardiness without the inclusion of dominance rule.

To investigate the effectiveness of the dominance rules, in particular, the impact on the runtime and total tardiness, for output data with the number jobs that occur in practice, the experiments are carried out. For this purpose the packets of 50 tasks are generated (instances, observations), each of 20 – 160 jobs, and the runtime of reducing the total tardiness estimated, using basic algorithm with and without the dominance rules, compared to algorithms without dominance rules, of local optimal solutions – attitude of obtained graph vertices in Hamilton path in a graph according to the rules, compared to the total amount of vertices (see Fig. 9, 10).

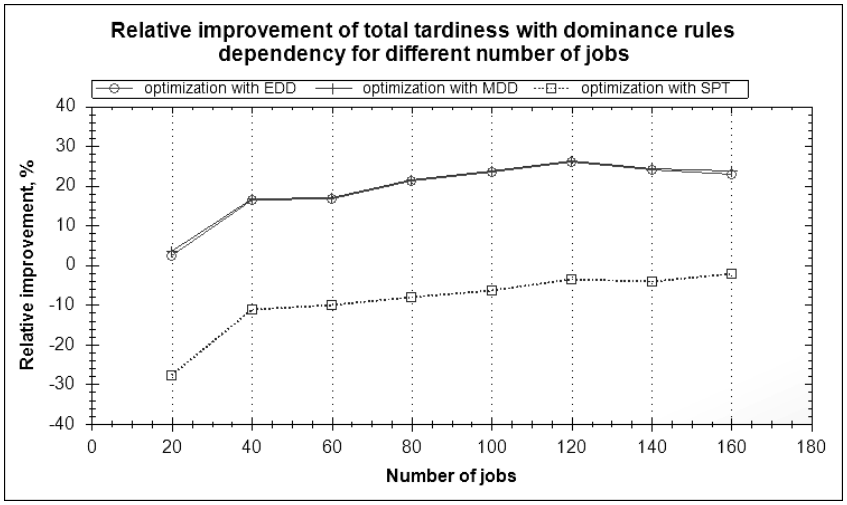

Fig. 9. The total tardiness relative reduction dependency using the dominance rules on the number of jobs, %

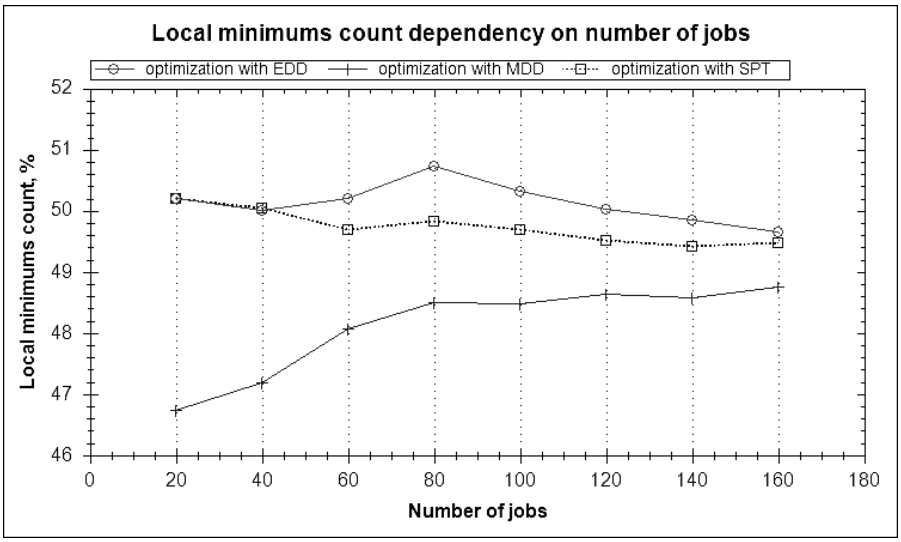

Fig. 10. Dependence of amount of local optimal solutions obtained by using the dominance rules, %

Conclusions

Analysis of the experimental results allows making the following conclusions regarding the effectiveness of the proposed algorithms using dominance rules for constructing optimal schedules of the tasks:

the inclusion of dominance rules to the basic algorithm does not increase the runtime under the algorithm, because in terms of time complexity, its effectiveness does not decrease;

using the dominance rules for this class of algorithms of minimizing the total tardiness allows to reduce the value of minimization criterion (total tardiness) in average (relative reduction – up to 20–25% for different dominance rules), which allows to conclude a significant efficiency of their use.

The future plans are to investigate the algorithms towards evaluating the effectiveness of the dominance rules based on the proposed metrics for different runtimes of jobs and the relationship between the runtimes and due dates.

1. C. Koulamas, The total tardiness problem: Review and extensions, Operations Research v.42 1994, p. 1025–1041.2. C. Koulamas, The single-machine total tardiness scheduling problem: Review and extensions, European Journal of Operational Research, v. 202 No.1 2010, pp. 1–7. 3 Tapan Sena, Joanne M. Suleka, Parthasarati Dileepan, Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey, Int. J. Production Economics, v.83 No. 1 2003, pp. 1–12.4. J. Du, J.Y.-T. Leung, Minimizing total tardiness on one machine is NP-hard, Mathematics of Operations Research, 1990 15, pp. 483–495. 5. E.L. Lawler, A ‘‘pseudo polynomial’’ algorithm for sequencing jobs to minimize total tardiness, Annals of Discrete Mathematics, No1 1977, pp. 331–342. 6. 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. 7. E.L. Lawler, A fully polynomial approximation scheme for the total tardiness problem, Operations Research Letters, No 1 1982, pp. 207–208. 8. 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. 9. M.Y. Kovalyov, Improving the complexities of approximation algorithms for optimization problems, Operations Research Letters, v. 17 March 1995, pp. 85–87. 10. 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. 11. Павлов А.А. Новый подход к решению задачи «Минимизация суммарного взвешенного опоздания при выполнении независимых заданий с директивными сроками одним прибором» / А.А. Павлов, Е.Б. Мисюра. // Системные исследования и информационные технологии. – 2002. – № 2. – С. 7–32. 12.Мінухін С.В. Метод мінімізації часу виконання завдань з директивними строками на некластеризованому ресурсі обчислювальної системи // Інформаційно-керуючі системи на

залізничному транспорті. – 2009. – №3. – С. 47 – 53. 13. S. Minukhin, Efficient method for Single machine total tardiness problem // IV International Conference «Problems of Cybernetics and Informatics» (PCI'2012), September 12–14, 2012. Електронний ресурс – режим доступу: www.pci2012.science.az/1/23.pdf. 14. Мінухін С.В. Дослідження методів мінімізації сумарного часу запізнювання робіт з директивними строками на одиночному обчислювальному ресурсі / С.В. Мінухін, Д.С. Лєнько // Проблеми і перспективи розвитку ІТ-індустрії. Системи обробки інформації. – 2013. – Вип. №3 (110). Т. 2. – С.30–35.15. Bean J. Accuracy of Modified date Rule / J. Bean, Hall D. Електронний ресурс – режим доступу: www.deepblue.lib.umich.edu.