Multi-machine Scheduling with Setup Times

Wojciech Bożejko[1(✉)], Łukasz Kacprzak[1], Piotr Nadybski[2], and Mieczysław Wodecki[3]

[1] Department of Automatics, Mechatronics and Control Systems, Faculty of Electronics, Wrocław University of Science and Technology, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
{wojciech.bozejko,lukasz.kacprzak}@pwr.edu.pl
[2] Faculty of Technical and Economic Science, The Witelon State University of Applied Sciences in Legnica, Sejmowa 5a, 59-220 Legnica, Poland
nadybskip@pwsz.legnica.edu.pl
[3] Institute of Computer Science, University of Wrocław, Joliot-Curie 15, 50-383 Wrocław, Poland
mieczyslaw.wodecki@uwr.edu.pl

Abstract. In this paper we are considering the problem of tasks scheduling executed simultaneously on multiple identical machines with a cost-criterion, which is the product of the tasks execution time and the number of used machines. Moreover, it is assumed that between the tasks performed sequentially there must be setup of the machines performed. Solution to the problem comes down to a generalization of a two-dimensional packing problem. The paper presents simulated annealing algorithm with different variants of packing strategy. The conducted computational experiments proved that designated solution differs little from some lower bounds for the constraints of the objective function.

1 Introduction

In the vast majority of the considered in the literature scheduling problems, it is assumed that a task is done on a single machine. However, in many real production processes, in particular - modern computational systems - a task execution requires the use of more than one machine (CPU), Bożejko et al. [5]. Then, we can talk about systems with parallel machines and multi-machine (multiprocessor) tasks. We can meet such an approach also in cloud exploration [17]. In literature, such problems have been considered for a long time, for instance in the work of Drozdowski [8,10] and in the monograph [9], Fleitelson [11], as well as in doctoral thesis of Kramer [13] where they mainly concern computer systems.

The first applications of multi-machine models of tasks scheduling refer not only to chemical industry (Bozoki and Richard [3]) but also to projects scheduling (Vizing [18]). The natural reference for this type of tasks are, examined from decades, multiprocessor computer systems (Błażewicz et al. [1]). In the vast majority of them the completion date of the execution of all tasks ($C_{\max}$) was maximized. Today, many computer centers offer performance of calculations

K. Saeed and W. Homenda (Eds.): CISIM 2016, LNCS 9842, pp. 300–311, 2016.
DOI: 10.1007/978-3-319-45378-1_27

and the cost of the service depends not only on the duration of the computations but also on the number of required processors. Therefore, there appears a need for the construction and testing of new, corresponding to the reality models and analysis of criterion functions taking into account the overall costs. We can encouter similar issues in construction projects planning, where the simultaneous use of multiple resources is the basis for the organization of work.

The basic problem of multi-machine scheduling, $P|size_j|C_{\max}$ (where P - is a symbol of multimachine, whereas $size_j$ – required number of machines) was studied mainly in reference to the approximate algorithms and the worst case scenario (Lin [15], Lloyd [16]). It has been proven that already in the case of two machines, the problem $P2|size_j|C_{\max}$ is *NP-hard*. Błądek et al. [2] consider the problem with $C_{\max}$ criterion assuming further that to certain tasks there must be assigned a fixed number of neighboring machines. In addition, it is assumed that the time of task scheduling undergoes learning or aging process, depending on the start of its execution.

In this paper we consider the problem of multi-machine scheduling, in which, between the tasks performed in sequence, there are additionally setups of machines introduced. In the case of construction processes, a setup is the time required for the movement of machines, equipment and employees. However, in computer systems, the setup time is related to the exchange (update) of the software transfer data/results, synchronization calculations, etc. Apart from the execution time of machine, the criteria, taking into account also other parameters (e.g. the number of used processors) undoubtedly enable more accurate determination of the actual cost of operating of such a system.

2 Problem Definition

The considered in the work problem of multi-machine tasks scheduling can be formulated as follows.

Problem: there is a set of tasks $\mathcal{J} = \{1, 2, \ldots, n\}$, given, which should be executed on identical machines (i.e. characterized with the same functional properties and equal capacities) from the set $\mathcal{M} = \{1, 2, \ldots, m\}$. Task $i \in \mathcal{J}$ requires at the same time execution of $size_i$ machines in $p_i > 0$ time units. Machines $l, k \in \mathcal{M}$ $(l \neq k)$ are called *neighbouring*, if $k = l+1$ $(l = 1, 2, ..., m-1)$ lub $k = l - 1$ $(l = 2, 3, ..., m)$. By $s_{i,j}$ we denote the time of machine setup after the completion of task i and before starting the task j $(i, j \in \mathcal{J})$, where both tasks are performed on the same machine. At the same time the following restrictions must be met:

(a) any machine, at any moment, cannot exercise more than one task,
(b) the task execution cannot be interrupted,
(c) each task is performed on the required number of neighboring machines,
(d) between the tasks executed sequentially there must be setup of the machine performed.

We assume further that the number of machines is $m \geqslant \max\{size_i\colon\ i \in \mathcal{J}\}$.

The considered in the work problem (in short denoted by **MPP**) boils down to determining, for each task, a subset of machines and the starting moments of its execution (for each machine) satisfying the constraints (a)–(d), to optimize the adopted criterion. The first solution can be represented by a pair of $\Theta = (Q, \mathcal{S})$ is such that:

- $\mathbf{Q} = [Q_1, Q_2, ..., Q_n]$, where Q_i ($Q_i \subseteq \mathcal{M}$, $|Q_i| = size_i$,) is a set of machines (assignment) on which the task $i \in \mathcal{J}$,
- $\mathcal{S} = [S_{1,1}, ..., S_{1,m}, S_{2,1}, ..., S_{2,m}, ..., S_{n,1}, ..., S_{n,m}]$, will be executed, where the element $S_{i,j}$ is the starting moment of task i execution on machine j (if $j \notin Q_i$, then we assume $S_{i,j} = -\infty$).

By Ω let us denote the set of all solutions to MPP problem.

For any solution $\Theta \in \Omega$

$$C_{\max}(\Theta) = \max\{S_{i,j} + p_i\colon\ \ i \in \mathcal{J}, j \in \mathcal{M}\}$$

is the moment of all task completion, and

$$M_{\max} = \max\{j\colon\ j \in Q_i,\ i \in \mathcal{J}\}$$

the maximum number of machine from all allocated to tasks execution. Let

$$F(\Theta) = C_{\max}(\Theta) \cdot M_{\max}(\Theta). \tag{1}$$

We consider the problem of scheduling of Textbf MPP tasks consisting in determination of solution $\Theta^* \in \Omega$ such that

$$F(\Theta^*) = \min\{F(\Theta)\colon\ \ \Theta \in \Omega\}.$$

Example. There is a set of multi-machine tasks given $\mathcal{J} = \{1, 2, 3, 4, 5, 6\}$ to be executed on $m = 4$ machines from the set $\mathcal{M} = \{1, 2, 3, 4\}$. The parameters of the individual tasks are shown in Table 1 (it is assumed that the setup times equal 0).

Table 1. Tasks parameters.

Task (j)	1	2	3	4	5	6
p_j	1	2	1	3	3	1
$size_j$	4	3	1	2	2	1

Figure 1 shows a Gantt diagram to certain solution $\Theta = (\mathbf{Q}, \mathcal{S})$, where $\mathbf{Q} = [Q_1, Q_2, Q_3, Q_4, Q_5, Q_6]$ and $Q_1 = \{1, 2, 3, 4\}$, $Q_2 = \{2, 3, 4\}$, $Q_3 =$

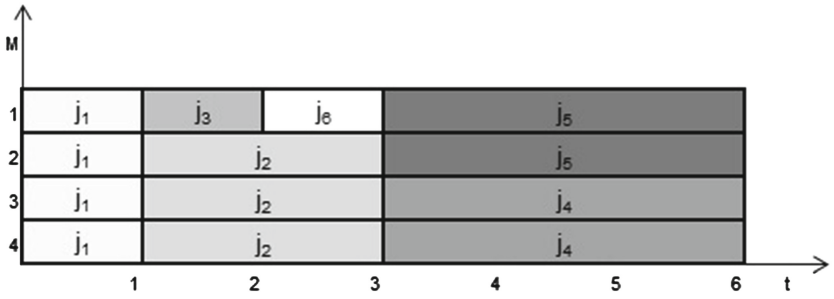

Fig. 1. Gantt diagram for solution Θ.

$\{1\}$, $Q_4 = \{3,4\}$, $Q_5 = \{1,2\}$, $Q_6 = \{1\}$. The beginning times of execution of tasks $\mathcal{S} = (0,0,0,0,-\infty,1,1,1,1,-\infty, -\infty,-\infty,-\infty,-\infty,3,3,3,3,-\infty,-\infty,2,-\infty,-\infty,-\infty)$. For this solution

$$M_{\max}(\Theta) = 4,\ C_{\max}(\Theta) = 6,$$

therefore the value of the goal function is $F(\Theta) = 24$.

The work [14] proved that the problem of multi-machine task scheduling (without setup times) is NP-hard. Some lower and upper bound estimation of the criterion (1) were presented in work [10]. To solve the considered in the work MPP problem there was a simulated annealing algorithm presented. The starting moments of tasks execution $\mathcal{S}$, at a fixed order, will be designated with the use of the algorithm solving two-dimensional packing problem (already one-dimensional problem of packing is strongly NP-hard, Garey et al. [12]).

3 Packing Problem

Two-dimensional bin packing problem consists in such arrangement of rectangles, that the field occupied by them had minimum surface area. In the considered variant the rectangles, which contact the sides (hights) should be moved from one other to some extent. To put the matter more formally, it can be described as follows:

Problem: there is a set of rectangles $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ given. The sizes of the rectangle $r_k \in \mathcal{R}$ are defined by a pair of numbers $t_k = (l_k, w_k)$, where l_k denotes the height, whereas w_k - width of the rectangle. By $d_{i,j}$ we denote the distance by which the rectangle r_i must be moved from r_j, when they are placed right next to one another.

The considered problem is to find the G shape of a minimal surface area, within which it is possible to place all rectangles, i.e. minimization of the product,

$$\Phi(G) = L \cdot W, \tag{2}$$

where L is the height and W - the width of G. shape. In short, the considered in the paper packing problem will be denoted by **BBP**.

With each rectangle located within the G shape there is identified the position, in which it was enclosed, represented by the coordinates in a Cartesian coordinate system. The rectangles should be placed inside the G shape in such a way so as not to overlap one other, whereas their edges, representing respectively their length and width, were located parallely to the edges representing the length and width of G. shape. The interval in the horizontal plane, between any two rectangles $r_i, r_j \in \mathcal{R}$ cannot be less than the value d_{r_i,r_j}. The example illustrating the above-described definitions and indications is given in Fig. 2.

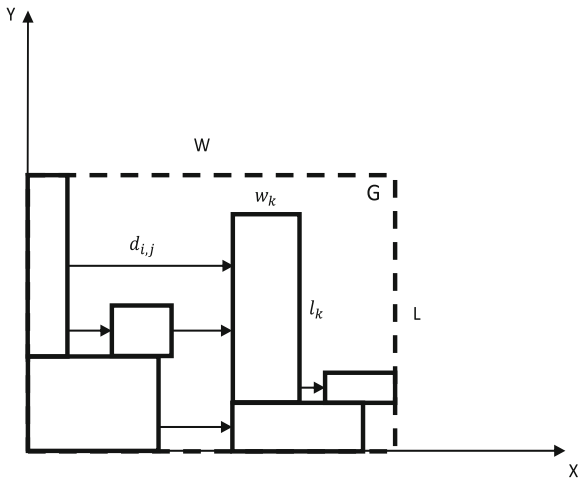

Fig. 2. G shape with enclosed rectangles

It is easy to observe the following relationship:

Corollary 1. *The problem of multi-machine task scheduling* MPP *is equivalent to the problem of two-dimensional bin packing* BBP.

3.1 Simulated Annealing Algorithm

The solution is represented in the form of a permutation (sequence) of rectangles

$$\alpha = [r_1, r_2, \ldots, r_n].$$

On its basis, using the **packing strategy** (the decoding solution), there is the value of the solution determined, i.e. the surface area of the $G(\alpha)$ shape in which rectangles are placed. Permutation Λ determines the order of rectangles in which they will be executed by the packing procedure. Let Ω be the set of all permutations of elements of $\mathcal{R}$, i.e. the set of feasible solutions. The simulated annealing algorithm (SA) will be used to search through this set.

The key elements of the algorithm are: initial temperature, probability function, with which the solutions worse than the current one, are accepted and cooling scheme (decreasing of the value) of the temperature.

In each step of the algorithm there are some elements the solutions space Λ - neighborhood of the current solution considered. It is a set of elements, generated by small modifications of the considered solution. In the case where the current solution is worse than the one drawn from the neighborhood, it is replaced by it. With some probability, it is also possible to replace the current solution with the worse solution. This strategy is used not only to prevent stagnation but also to direct the trajectory of explorations in new areas of the solution space. The selection of temperature, cooling scheme and the probability function has influence on the frequency of accepting solutions worse than the current solutions, thus not only on the ability of the algorithm to leave local minima but also on the stability of the search itself.

Let $\alpha \in \Omega$ be any starting solution. By $\mathcal{N}(\alpha)$ we denote the neighborhood, whereas by t the control parameter (temperature).

Simulated annealing algorithm

$\alpha_{best} \leftarrow \alpha$;
`repeat`
  `repeat`
    Randomly determine element β from neighborhood $\mathcal{N}(\alpha)$;
    `if` $G(\beta) < G(\alpha)$ `then` $\alpha \leftarrow \beta$
    `else`
      `if` $exp\{-(G(\beta) - G(\alpha))/t\} > random$ `then` $\alpha \leftarrow \beta$;
    `if` $G(\beta) < G(\alpha_{best})$ `then`
      $\alpha_{best} \leftarrow \beta$;
  `until` *Change_control_parameter*;
  Change control parameter t;
`until` *End_Condition.*

Neighborhood $\mathcal{N}(\alpha)$ is generated by two types of random moves, namely - the swap and the twist type. Such neighborhoods are described in details in the works of Bożejko and Wodecki [4,6,7]. The temperature, in turn, is changed according to the geometric distribution, i.e. $t_{k+1} = t_k \cdot \lambda$ where lambda $0,8 \leqslant \lambda < 1$.

3.2 Packing Strategy

In the construction of rectangle packing procedures, i.e. in the process of determination of the shape (area) within which the rectangle is enclosed, there are different strategies applied. They differ from one another in the criteria for selecting the area (coordinates), in which the considered rectangle will be placed. In each of the used strategies we consider rectangles in the order they appear in the solution (permutation), at the same time trying to designate the space of the smallest possible surface area. A general description of the rectangle packing procedure is presented below.

Let $\alpha = [r_1, r_2, \ldots, r_n]$ be a certain sequence (permutation) of rectangles. The rectangles are considered in the order they appear in α. In i-th iteration of the procedure ($i = 1, 2, \ldots, n$) we will enclose the rectangle r_i to G shape (area), in which there are already enclosed rectangles $r_1, r_2, \ldots, r_{i-1}$. The set $P_{i-1} = \{z_1, z_2, \ldots, z_s\}$, where $z_i = (x_i, y_i)$ contains positions on which the considered rectangle is inserted (we assume that at the starting moment the area $G = \emptyset$ and $P_0 = \{(0, 0)\}$). Next, rectangle r_i is 'temporarily' placed into the G, area (shape) on each position from the set P_{i-1}. In this way, we generate a set of temporary areas $O = \{G_1, G_2, \ldots, G_s\}$. Depending on the used packing strategy, we choose from O corresponding area (i.e. the coordinates of the point where we insert a rectangle r_i). If the area G_l, was chosen, then:

(i) rectangle r_i is placed into the G shape on the position $z_l = (x_l, y_l)$,
(ii) the set of positions $P_i = P_{i-1} \setminus \{z_l\} \cup \{(x_l + d_{r_c,r_i} \cdot \delta, y_l), (x_l + d_{r_c,r_i} \cdot \delta, y_l + l_i)\}$, where $\delta \in \{0, 1\}$ and takes the value 1, if the placed rectangle r_i was moved by size d_{r_c,r_i}, and position z_l was created in the result of placing the rectangle r_c. Otherwise $\delta = 0$.

Next, we move to the next iteration. The procedure ends after placing all rectangles on their positions. When selecting position, in which the following rectangle is inserted, the following strategies were used:

1. *Minimum area* - the position, causing minimal enlargement of the area (2) is chosen, created as a result of the insertion of the considered rectangle,
2. *Area-sizes* - it is the development of the minimum area strategy. The value of the ratio is calculated, taking into account the surface area of the temporary (area) shape and the difference in the length of its sides. The purpose of application of this factor is to create areas having a uniform shape,
3. *Roulette* - the position is drawn in accordance with the principle of a roulette. The probability of drawing is proportional to the size of the temporary area,
4. *XYZ* - similarly as in the case of a roulette but the drawing is carried out according to the uniform distribution.

In the simulated annealing algorithm, each of the above-described packing strategy was implemented. For each designated by the algorithm solution G there was *packing accuracy* calculated $T/\Phi(G)$, where T is the sum of all the areas of rectangles (the lower bound of function Φ).

4 Computational Experiments

The computations were performed on a computer equipped with a 6-core Intel Core i7 X980 CPU (3.33 GHz) and Linux operating system Ubuntu 12.04.5 LTS. For the purposes of computational experiments there were test data of varying degrees of heterogeneity randomly generated. The number of possible types of rectangles is 3(107), 5(101), 8(143), 10(134), 12(142), 15(129), 20(161). The total number of rectangles is given in brackets. The following values of the SA algorithm parameters were adopted: initial temperature $t_0 = 125$, rate of temperature change $\lambda = 0,98$, while the temperature change occurs every 100 iterations

(return to the initial value every 500 iterations) and a maximum number of iterations is 5000. The results of the carried out study are presented in Tables (2, 3, 4 and 5). Each row of the table contains the results obtained for one installation of test data: the number of types of rectangles (*Types*), the accuracy of the designated solution (*Accuracy*), the surface area of the shape (area) including the considered rectangles (*Shape area*), lengths of the shorter and the longer side of the surface area (*Shorter, Longer side*), the ratio of the longer side to the shorter side(*Ratio*) and the calculations time in seconds (*Time*). By far the best results for all instances of the problem were obtained by applying the strategy of minimum area (Table 2) in the packing procedure. For each example there was the shortest time of calculations observed. The parameter values *Ratio* were much greater than the value received after the application of other packing strategies (longer side was from 12.23 to 60.00 times larger than the shorter side). This strategy shows a clear preference of the surface area of the specified space. Using the Area-sizes ratio strategy (Table 3) the results were only a little worse than those listed in the Table 2. With the use of this strategy the smallest value of the *Ratio* coefficient for the number of types 3, 8, 10, 12, 15 and 20 were obtained. For the two types 3 and 5, the values were the same as in the strategy Roulette strategy. The value of the *Ratio* parameter close to 1.00 means that shape of the designated area is close to the square.

Table 2. Minimum area strategy.

Types	Accuracy	Shape area	Shorter side	Longer side	Ratio	Time
3	0.920	489176	94	5204	55.36	56
5	0.873	677100	150	4514	30.09	37
8	0.850	766140	113	6780	60.00	117
10	0.860	641200	229	2800	12.23	162
12	0.842	689475	145	4755	32.79	171
15	0.874	699470	226	3095	13.69	102
20	0.841	892080	216	4130	19.12	160

Table 3. Area-sizes ratio strategy.

Types	Accuracy	Shape area	Shorter side	Longer side	Ratio	Time
3	0.861	522440	706	740	1.05	141
5	0.851	694540	820	847	1.03	115
8	0.832	782310	879	890	1.01	278
10	0.803	686412	828	829	1.00	217
12	0.816	711490	842	845	1.00	254
15	0.819	746496	864	864	1.00	190
20	0.804	933131	961	971	1.01	376

Table 4. Roulette strategy.

Types	Accuracy	Shape area	Shortest side	Longer side	Ratio	Time
3	0.220	2042350	1396	1463	1.05	142
5	0.303	1947150	1379	1412	1.02	121
8	0.183	3565740	1774	2010	1.13	317
10	0.256	2151000	1434	1500	1.05	268
12	0.193	3009260	1704	1766	1.04	315
15	0.201	3043580	1689	1802	1.07	238
20	0.148	5069220	2075	2443	1.18	444

Table 5. XYP strategy.

Types	Accuracy	Shape area	Shorter side	Longer side	Ratio	Time
3	0.134	3352340	1713	1957	1.14	180
5	0.244	2423280	1366	1774	1.30	154
8	0.157	4158810	1622	2564	1.58	423
10	0.182	3035430	1305	2326	1.78	354
12	0.167	3476110	1547	2247	1.45	417
15	0.168	3632800	1353	2685	1.98	315
20	0.151	4966610	1793	2770	1.54	604

The next two Tables 4 and 5 include the results obtained after application of non-deterministic packing strategy in SW algorithm. The values of the solutions obtained with the use of the Roulette strategy (Table 4) were significantly worse than ones obtained with the use of Minimum area or area-size ratio strategy. Also, the time of computations was longer. For the two types 3 and 5 there were the biggest values of the *Ratio* parameter obtained.

While applying the strategy XYP (the results - Table 5) there were the worst solutions, for the six types, obtained. This concerned both the computation time and the accuracy. Generally, deterministic strategies proved to be much better than probabilistic ones. If we take into account only the surface area, it is better to use the Minimum area strategy. On the other hand, when we are looking for an area of shape similar to a square, then it is better to use the Area-sizes strategy. For the determination of costs of multiprocessor tasks, the choice of packing strategy, depends on the relationship between the time of CPU utilization (computations time) and the number of processors used.

Simulated Annealing and Genetic Algorithm Comparison. The canonical version of the Genetic algorithm was adopted. The algorithm has been configured as follows. For crossover, the PMX method was used, the Mutation was based on random chromosome fields exchange. The stopping condition was the maximum iteration number – 5000.

Table 6. Simulated annealing and genetic algorithms comparison

	Simulated annealing			Genetic algorithm		
	Accuracy	Ratio	Time	Accuracy	Ratio	Times
Area	0.920	55.36	55.7	0.920	55.36	15
Area-sizes	0.861	1.05	140.9	0.894	1.05	16
Roullette	0.220	1.05	141.7	0.256	1.16	149
XYP	0.134	1.14	180.0	0.323	1.86	34
Area	0.873	30.09	36.5	0.872	58.11	21
Area-sizes	0.851	1.03	114.8	0.892	1.00	23
Roullette	0.303	1.02	121.1	0.346	1.22	130
XYP	0.244	1.30	154.3	0.462	1.24	37
Area	0.850	60.00	117.4	0.898	15.31	30
Area-sizes	0.832	1.01	278.1	0.866	1.00	63
Roullette	0.183	1.13	317.4	0.236	1.44	345
XYP	0.157	1.58	423.3	0.313	1.78	88
Area	0.860	12.23	161.8	0.887	11.61	44
Area-sizes	0,803	1.00	217.1	0.845	1.00	44
Roullette	0.256	1.05	268.3	0.305	1.04	322
XYP	0.182	1.78	354.2	0.418	1.22	69
Area	0.842	32.79	170.9	0.903	20.30	54
Area-sizes	0.816	1.00	254.4	0.852	1.00	52
Roullette	0.193	1.04	315.2	0.261	1.04	340
XYP	0.167	1.45	416.8	0.356	1.24	89
Area	0.874	13.69	101.6	0.903	13.79	36
Area-sizes	0.819	1.00	190.3	0.851	1.01	44
Roullette	0.201	1.07	238.3	0.306	1.03	280
XYP	0.168	1.98	315.0	0.390	1.33	60
Area	0.841	19.12	160.1	0.853	9.58	54
Area-sizes	0.804	1.01	376.1	0.835	1.01	91
Roullette	0.148	1.18	443.5	0.218	1.07	489
XYP	0.151	1.54	603.8	0.266	1.64	132

There are Accuracy, Ratio and Time results, for all strategies combined with Simulated Annealing and Genetic Algorithm compered in Table 6. The results are organized in four-row data structures, for all test data instances (3, 5, 8, 10 12, 15 and 20 types). Each row presents results for particular strategy and data set. The usage of Genetic algorithm in most cases improved the Accuracy results, especially for XYP strategy. Thirteen times the Ratio had greater value when Simulated Annealing algorithm was used, six times the result was equal. This

means that structures builded with Genetic Algorithm had the shape close to square more often. The computation time was shorter in all cases for Area, Area-sized and XYP strategy combined with Genetic Algorithm. Roulette packing strategy had shorter execution time with Simulated Annealing than with Genetic Algorithm.

5 Conclusions

In the paper there was the problem of multi-machine tasks scheduling with setup times and minimizing the product of the number of machines used and the length of time of all tasks execution considered. Designation of a solution can be brought down to a two-dimensional rectangles packing problem with minimization of the surface area of the occupied area (shape). The simulated annealing algorithm using different packing strategies was presented in the work, i.e. placing rectangles within the proposed area. The conducted computational experiments showed that the designated surface area (to solve the tasks scheduling problem) are only slightly greater than the sum of surface areas of all the rectangles (i.e. the lower bound).

References

1. Błażewicz, J., Drabowski, M., Węglarz, J.: Scheduling multiprocessor tasks to minimize schedule length. IEEE Trans. Comput. **C–35**(5), 389–393 (1986)
2. Błdek, I., Drozdowski, M., Guinand, F., Schepler, X.: On contiguous and non-contiguous parallel task scheduling. J. Sched. **18**, 487–495 (2015)
3. Bozoki, G., Richard, J.-P.: A branch-and-bound algorithm for the continous-process job-shop scheduling problem. AIIE Trans. **2**(3), 246–252 (1970)
4. Bożejko, W., Wodecki, M.: On the theoretical properties of swap multimoves. Oper. Res. Lett. **35**(2), 227–231 (2006)
5. Bożejko, W., Gniewkowski, Ł., Wodecki, M.: Solving timetabling problems on GPU. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014, Part II. LNCS, vol. 8468, pp. 445–455. Springer, Heidelberg (2014)
6. Bożejko, W., Wodecki, M.: Solving permutational routing problems by population-based metaheuristics. Comput. Ind. Eng. **57**, 269–276 (2009)
7. Bożejko, W., Wodecki, M.: Parallel genetic algorithm for the flow shop scheduling problem. In: Wyrzykowski, R., Dongarra, J., Paprzycki, M., Waśniewski, J. (eds.) PPAM 2004. LNCS, vol. 3019, pp. 566–571. Springer, Heidelberg (2004)
8. Drozdowski, M.: Scheduling multiprocessor tasks an overview. Eur. J. Oper. Res. **94**, 215–230 (1996)
9. Drozdowski, M.: Select Problems of Scheduling Tasks in Multiprocessor Computer System. Poznań University of Technology Press, Poznań (1997). Series: Monographs, No. 321
10. Drozdowski, M.: Scheduling for Parallel Processing. Computer Communications and Network. Springer, Berlin (2009)

11. Feitelson, D.G., Rudolph, L., Schwiegelshohn, U., Sevcik, K.C., Wang, P.: Theory and practice in parallel job scheduling. In: Feitelson, D.G., Rudolph, L. (eds.) IPPS-WS 1997 and JSSPP 1997. LNCS, vol. 1291, pp. 1–34. Springer, Heidelberg (1997)
12. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
13. Kramer, A.: Scheduling multiprocessor tasks on dedicated processors. Ph.D. thesis, Fachbereich Mathematik/Informatik, Univeristat Osnabruck (1995)
14. Lee, C.Y., Cai, X.: Scheduling one and two-processor tasks on two parallel processors. IIE Trans. **31**(5), 445–455 (1999)
15. Lin, J., Chen, S.: Scheduling algorithm for nonpreemptive multiprocessor tasks. Comput. Math. Appl. **28**(4), 85–92 (1994)
16. Lloyd, E.: Concurrent task system. Oper. Res. **29**(1), 189–201 (1981)
17. Smutnicki, C., Pempera, J., Rudy, J., Żelazny, D.: A new approach for multi-criteria scheduling. Comput. Ind. Eng. **90**, 212–220 (2015)
18. Vizing, V.: About schedules observing deadlines. Kibernetika **1**, 128–135 (1981)