INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2024, VOL. 70, NO. 1, PP. 59–69
Manuscript received November 14, 2023; revised March, 2024. doi: 10.24425/ijet.2024.149515

Optimization of a task schedule for teams with members having various skills

Marek Bazan, Czesław Smutnicki, and Maciej E. Marchwiany

***Abstract*—We consider the real-life problem of planning tasks for teams in a corporation, in conditions of some restrictions. The problem takes into account various constraints, such as for instance flexible working hours, common meeting periods, time set aside for self-learning, lunchtimes and periodic performance of tasks. Additionally, only a part of the team may participate in meetings, and each team member may have their own periodic tasks such as self-development. We propose an algorithm that is an extension of the algorithm dedicated for scheduling on parallel unrelated processors with the makespan criterion. Our approach assumes that each task can be defined by a subset of employees or an entire team. However, each worker is of a different efficiency, so task completion times may differ. Moreover, the tasks are prioritized. The problem is NP-hard. Numerical experiments cover benchmarks with 10 instances of 100 tasks assigned to a 5-person team. For all instances, various algorithms such as branch-and-bound, genetic and tabu search have been tested.**

***Keywords*—unrelated parallel machine scheduling problem; task scheduling for corporation teams; genetic algorithms; tabu search; branch and bound**

I. Introduction

THE Task scheduling for teams with diverse skill sets, working hours, and fixed activities is not a trivial challenge for any team and manager. The optimization of task schedules benefits the organization, employees, and clients alike. It reduces project implementation time, lowers costs, and mitigates stress within the team. In this paper we describe a solution of a scheduling problem with the following real-life assumptions:

1) there are M members of the team,
2) there are N tasks that may be executed by various team members
3) the m-th ($m \in \{1, \dots, M\}$ team member executes the n-th task with a time t_{nm} ,
4) tasks have their priorities such that tasks with higher priorities have to be executed first.

This work was financed with European Union funds from the Smart Growth Operational Programme 2014-2020, Measure 1.1.: R&D projects of enterprises. Sub-measure 1.1.1.: Industrial research and development carried out by enterprises.

M. Bazan and Cz. Smutnicki are with Wroclaw University of Science and Technology, Department of Computer Engineering, Wrocław, Poland (e-mail: marek.bazan@pwr.edu.pl, czeslaw.smutnicki@pwr.edu.pl).

M. Bazan and M.E. Marchwiany are with JT Weston sp. z o.o. Warszawa, Poland (e-mail: marek.bazan@pwr.edu.pl, maciej.marchwiany@jtweston.pl).

The objective is to find an assignment of tasks to the team members so that the makespan of tasks within a week is the least with the additional constraints that

1) groups of a certain number or all team members should be synchronized on meetings during a workday,
2) each team member may work hours specific to him/her,
3) it should be possible to plan cyclical tasks specific to each team member or to a group of members (such as lunch breaks, self-development hours, foreign language lessons etc.)

The important assumption about the executed tasks is that each task may by interrupted by a meeting or the end of work for a particular weekday.

The proposed method of solving the problem stated at the beginning of the paper consists of two phases. At first, we show how to define this problem as a $R||C_{max}$ [1] i.e., scheduling on unrelated parallel machines problem, where machines correspond to team members and the execution times for a specific task for various team members may model the skills of each member. The minimization of the completion time of the task, which is completed as the last, provides the makespan minimization whereas sorting of the obtained solution according to their priorities for each team member allows preserving the task prioritization. Next, the schedule obtained from $R||C_{max}$ problem is "cut" for groups of team members to ensure the synchronization of groups of team members at meetings. Additionally, cutting and expanding a schedule allows us to model flexible working hours during a week and to plan cyclical tasks such as self-development hours, lunch breaks and in general flexible working hours.

The remainder of the paper is organized as follows. The second section presents related work concerning algorithms to solve unrelated machine scheduling problems. The third section presents a proposed formulation of the problem. The next section describes three methods to solve the problem, classical and grouping genetic algorithms with several genetic operators for crossover and mutation with grouping operators for initial population generation, crossover and mutation, Tabu Search method and finally, Branch and Bound method is described.

II. Related Work

Task scheduling is widely covered in the literature. However, authors have primarily focused on scheduling jobs in queue systems for HPC (High Performance Computing) clusters and clouds [2]–[5], or scheduling work in factories [6]–[8]. There is limited information in the literature regarding task scheduling for human teams. One can find methodologies for scheduling tasks for multiple robots [9], [10] or medical equipment scheduling [11], the domain of human work scheduling remains relatively unexplored. One commonly adopted approach in human work scheduling is the 'shop of tasks' methodology [12], [13]. However, this approach does not fit in corporate workflow with ticket systems and modern efficiency-focus team management.

Task-scheduling methods dedicated to an NP-hard problem are diverse. The most commonly mentioned in this aim are:

- genetic algorithm [14]–[16],
- particle swarm optimization [17]–[20],
- ant colony optimization algorithms [21], [22],
- multi-processors-based algorithms [23],
- fuzzy algorithms [24], [25],
- cost-based algorithms [26].

Further, we implement only a few of these approaches.

III. Theory

A. Proposed Method

In this section, we describe the adopted assumptions of the model of organizing the work of the team and its formulation as a job-scheduling problem:

1) The task may be stopped and resumed.
2) The task may not be stopped unfinished and handed to another team member.
3) Each team member executes his/her tasks according to a sequence of priorities.
4) The execution time of each task depends on the team member who does it, which enables the modeling of different skill levels to perform a given task.

It is worth noting that we avoid planning cyclical tasks as tasks that are scheduled by the algorithm. They are planned by cuts and shifts of the obtained schedule for any group of team members.

The optimization criteria is the length of the schedule, which is the maximum of an individual team member makespan, which is the minimization of $C_{\max}$ criterion. The usage of this criterion provides a load balance for all team members and also favors assigning a task to more competent team members, i.e., to those having a shorter execution time for a given task.

For N tasks and M team members the problem may be formulated as a linear programming problem

$$\min_{\mathbf{x}\in\mathbf{R}^{N\times M}} C_{\max} \tag{1}$$

with constraints

$$z_m + \sum_{n=1}^{N} x_{nm}t_{nm} \leq C_{\max}, \quad m \in \{1,\ldots,M\}, \tag{2}$$

$$\sum_{m=1}^{M} x_{nm} = 1, \quad n \in \{1,\ldots,N\}, \tag{3}$$

$$x_{nm} \in \{0,1\},\ n \in \{1,\ldots,N\}, m \in \{1,\ldots,M\}, \tag{4}$$

where t_{nm} is the time required to execute the task n-th by the m-th team member. The constraint (2) shows the load of the m-th team member with tasks. The constraints (3) and (4) correspond to the condition that the whole task may be executed by only one team member.

The problem (1) – (4) is a mixed linear programming task. It may be solved using the simplex method (see, e.g., [27]) or the interior point method (see e.g., [28]) for a small N and M. The problem can also be perceived as a task scheduling on unrelated parallel machines with the makespan criterion. The problem is NP-hard (see e.g., [29]), which inclines researchers to find approximation schemes. A short overview of the results in this field can be found at [30]. Currently, the best approximation scheme with quality $(1+\epsilon)$ for the problem has the following complexity: $N(M/\epsilon)^{O(M)}$, [30]. For a constant number of machines, it provides a fully polynomial-time approximation scheme, which computes for any fixed $\epsilon > 0$ an ϵ-approximate solution in $O(N)$ time. In a general case, the complexity versus quality of the scheme is still not competitive with metaheuristics.

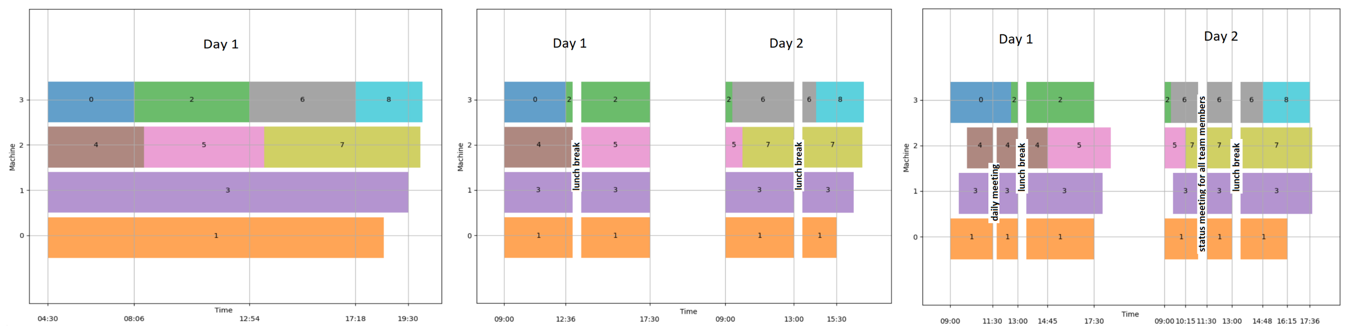
After solving problem (1) – (4) the final solution is generated in the following steps:

1) Extract tasks assigned to m-th team member,
2) Sort the extracted tasks according to priorities p_n in descending order.
3) Construct a schedule that is shifted to the left on a timeline. If a team member has an individual starting time for a certain day or has some remaining tasks from the previous day, then start from the earliest possible starting time.
4) Cut each individual schedule according to working hours in chosen days for each individual team member.
5) Synchronize cut blocks among team members so that meetings start at the same time for all participants.

The overall approach is visualized in Figure 1 with the simple example solved in [31]. A solution of an unrelated machine scheduling problem is shown in Figure 1a. The optimal solution lasts 15.6 hours and is plotted as one full day. The execution will take almost two working days for the team members. Figure 1b shows a two-day execution with additional lunch breaks, achieved by cuts and shifts of the initial solution. Figure 1c shows an example with additional team meetings (for part of and the whole team). In Figures 1b and 1c, the tasks when meetings or planned breaks begin are clear for the reader.

B. Load balancing

The presented approach up till now does not work well when the amount of time foreseen for meetings during a week is not equal for all team members. If we count meeting time as time at work, then employees with the highest duration of meetings in the schedule work the longest, since meetings are not added in the $C_{\max}$ objective function but applied using cuts and shifts after the optimization process is finished. One



(a) The optimal solution - 15.6 hours, placed in one full day (24h)

(b) Solution stretched to two days with lunch breaks

(c) Solution stretched to two days with lunch breaks, daily and status meetings

Fig. 1: An example solution for 4 machines and 9 tasks from [31]. The total (optimal) time is $C_{\max} = 15.6$ hours – two working days. (a) The initial solution placed within one working day. (b) The solution cut and shifted according to the availability mask of the employees. (c) The solution with personalized working hours of the employees and additional meeting cuts and shifts.

can say that optimization balances only core working time spent on task execution – see Figure 2a.

The second issue to be resolved is balancing the workload when one or more team members works only part-time. In such a case, the algorithm allocates too much work for such an employee. The example of such a situation is shown in Figure 3a.

The solution in both of the aforementioned cases is adding virtual tasks. In the case of unequal time spent on meetings in a week, one adds virtual tasks for team members with such long execution time as their amount of time spent on meetings surpasses the time spent on meetings by a team member with the shortest overall meeting time during a week. These virtual tasks have to have the shortest execution time for the team member to be chosen in the final timetable so that this particular task is performed by this member. An example of this solution is presented in Figure 2b.

The solution in the case of unequal work duration during the week, i.e., some employees not working full-time, is analogue. A virtual task is added to a team member who works part-time. The duration of the execution of this task is for this team member set equal to the difference of full-time working and part-time working. An example of the balanced schedule is presented in Figure 3b.

C. Optimization algorithms and feasibility of a solution

To solve our problem we use the following algorithms:

1) Genetic algorithm [32] with various crossover and mutation operators [33],
2) Tabu Search [34],
3) Exact method based on the branch and bound approach presented in [35].

We encoded the solution which for the main formulation Eq. (1)-(4) is given by the matrix $X = [x_{nm}]_{1 \leq n \leq N, 1 \leq m \leq M}$ to a vector $s = [s_1, \ldots, s_N]$ where $s_n := m$ and m is an index of the machine on which the n-th task is executed. The advantage of such a representation is that as soon as $s_n \in [1, \ldots, m]$ for $1 \leq n \leq N$ the solution is feasible. As one can see, this representation does not store information on the sequence of execution of tasks by an employee. If any sequence of tasks execution is required, it has to be modeled by the vector of priorities. Some details of the investigated algorithms are given below. Note that sorting of the single employee tasks execution order with respect to any priority will preserve the feasibility of the solution. Moreover, the process of cuts and shifts of tasks (described at the end of Section III-A) also still maintains feasibility. This follows from the fact that a task may be processed for a given time after which it can be stopped and after a break, it is resumed.

To check if all meetings are within time slots when all interested participants are available in the office is independent of an optimization algorithm used. It is done beforehand.

The following optimization algorithms are then investigated:
Genetic algorithms. We have tested two procedures for generating the initial population. First, this was a usual random generation of the machine to execute the n-th task. The second approach used initialization based on `min` operator from the genetic grouping algorithm [33].

As a crossover operator, we present the results for four different operators, such as

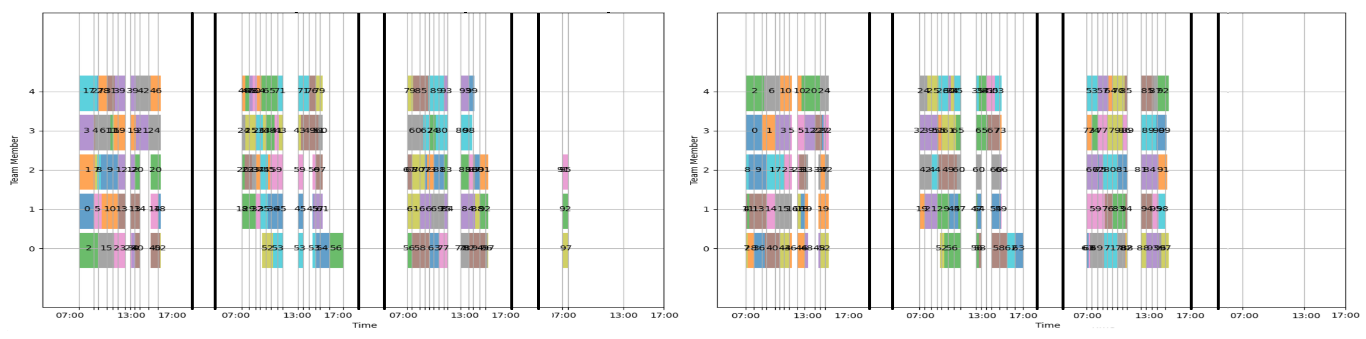
- `min` from [33] typical for grouping genetic algorithms,
- `MSXF` from [36],
- `2points` which is similar to `PMX` [37] but two offsprings are created by swapping a random subsequence and leaving the rest unchanged, transferred from parents since we do not work on permutations.
- `OBX` from [38].

Crossover probability was set to $p_{crossover} = 0.8$.

As a mutation operator, we used the „`download`” mutation operator from [33] and a usual mutation performed by a randomized change in a prefined number at randomly chosen positions from one team member to another. Mutation probability was set to $p_{mutation} = 0.2$.

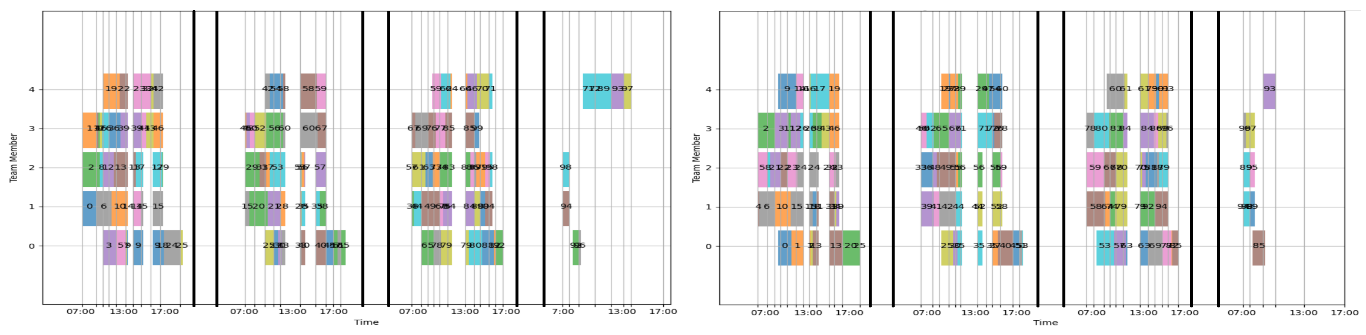
The survival mechanism was based on choosing the best individuals out of offsprings and parents to form a new generation.

The population size is set to 50. The number of iterations was set to 200.



(a) unbalanced tasks durations (b) balanced tasks durations

Fig. 2: Visualizations depicting task scheduling solutions for 100 tasks among 5 team members. The left-hand picture (a) optimizes makespan ($C_{\max}$), in the right-hand picture (b), task durations are balanced.



(a) unbalanced tasks durations (b) balanced tasks durations

Fig. 3: An example of scheduling 100 tasks for 5 team members with unbalanced and balanced task duration. Team member number 4 works 3/4 of the full-time but gets approximately the same amount of tasks to perform. This is an unwanted behavior.

Tabu Search The tabu search that we employed is based on [34]. To generate a starting point, we use `min` operator [33] used to produce the initial population. However, we do not use the best solution but the one closest to the median. It comes from the observation that the `min` operator often tends to generate a solution that is close to a certain local minimum. When the optimization process is started from the solution corresponding to the one with the objective function close to the median, it gave us better results than genetic algorithms, since the convergence to a lowr objective function value was observed more often than for the genetic algorithm. For the operators that are used to generate a solution in the neighborhood, the following operators were used:

1) `NH1` – demands that the tasks to be swapped are executed by another team member,
2) `NH2` – swaps tasks performed by different team members if at least one of them has a different processing time when executed by another team member,
3) `NH3` – swaps tasks performed by different team members if a swap improves the sum of the processing time for these team members.

Our implementation does not include a combination of the above operators. The number of iterations is set to 1000, and the number of tries in the neighborhood is set to 150. Note that all operators allow hill climbing since the conditions cover only two machines.

Exact algorithm The exact algorithm that we have used to calculate optimal solutions [35] is a branch and bound algorithm for unrelated parallel machines using Lagrangian relaxation suggested in [1]. The instances we exploited to test the overall approach to schedule tasks for a team of employees come from [35] and are hard for mixed-integer linear programming algorithms – see [39]. The source code published by the authors of [35] was exploited to find exact solutions for the instances used for presentation in this paper.

IV. Experiments

The aim of the experiments was to compare the quality of the solutions obtained by various algorithms. We consider 10 test instances with 100 tasks to be scheduled for 5 employees in the team. The instances were taken from [35]. The times given in [35] are treated by us to be given in minutes. In all tables in the present paper the values in the objective function as well as inprovements are given in hours. On the solution, we put the mask of team members' availability – i.e., individual working hours as well as meetings that are synchronized to all or a chosen group of employees. The availability of team members considered in this paper is given in Table I and meetings are shown in Table II. The example of some correct solutions are shown in Figures 2b, 3b and 4.

TABLE I: Working hours for team members

Day	Empl. 0	Empl. 1	Empl. 2	Empl. 3	Empl. 4
Mo	7:00-15:00	7:00-15:00	7:00-15:00	7:00-15:00	7:00-15:00
Tu	9:00-17:00	7:00-15:00	7:00-15:00	7:00-15:00	7:00-15:00
We	7:00-15:00	7:00-15:00	7:00-15:00	7:00-15:00	7:00-15:00
Th	7:00-15:00	7:00-15:00	7:00-15:00	7:00-15:00	7:00-15:00
Fri	7:00-15:00	7:00-15:00	7:00-15:00	7:00-15:00	7:00-15:00

Note that the availability for a given employee may be represented by multiple intervals during the day as well – see Figure 4. In the availability table, Employee number 2 begins work at 9:00 on Tuesday and completes an 8-hour shift.

TABLE II: Meetings plan for a week

Day	Time	Title	Participants
Mo	11:30-12:00	Daily	Employees 0,1,2,3,4
	13:00-14:00	English classes	Employees 0,1,2
Tu	11:00-12:30	Meeting	Employees 0,1,2,3,4
	13:00-14:00	English classes	Employees 0,1,2
We	11:00-12:15	Status 2	Employees 0,1,2

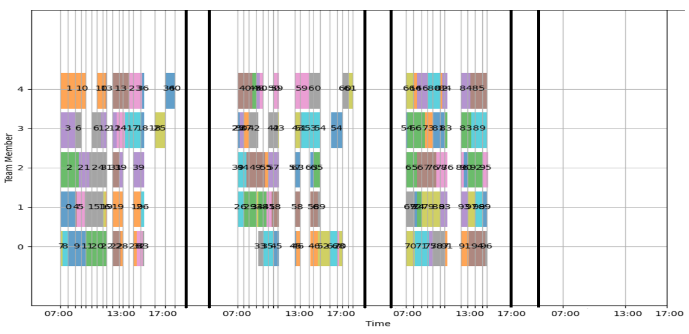


Fig. 4: An example solution of scheduling of 100 tasks for 5 team members. The team members with numbers 3 and 4 on Monday and Tuesday work in three different time slots, respectively. Such slots have to be declared by the employees before generating the schedule.

We tested three algorithms to solve the $R||C_{max}$ problem to which our scheduling problem is reduced: a genetic algorithm with various crossover, mutation and initial population generation operators, tabu-search with different neighborhood operators, and the branch and bound algorithm. Results are presented in Tables III and IV for the genetic algorithm, in Table V for tabu-search, and in Table VI for the branch and bound algorithm. For the branch and bound algorithm, the optimal schedule values were found. Only in instances 2 and 5, does the makespan exceeded three days i.e. it is greater than 72 hours (c.f. Table VI).

The tables for the genetic algorithm present the maximum and mean differences in total execution time compared with the achieved minimum for a selected setup. The results were computed for all instances with 5 repeated runs (c.f . Tables III and IV). Detailed results can be found in the Appendix. As can be seen from Tables XVII-XXVI in the Appendix, the three-day makespan was achieved for instances 1,7,8 and 9.

The conclusion is that the sophisticated initial population operators (like `min` operator) generate a fairly good solution. The metaheuristic algorithm does not significantly improve the initial state. However, the unsophisticated initial population (such as a random initialization) can be easily improved. In all the cells where the value is greater than 12 hours, it means that the initial schedule spanned to 4 days, and after the optimization, the makespan was shortened to 3 days. It is worth noting that all improvements that are greater than 1.75 hours mean more than one man-day of savings since the schedule is balanced for 5 employees.

The table for the tabu search algorithm presents analogously the maximum and mean differences in total execution time compared with the achieved minimum for a selected setup. The results were computed for all instances with 5 repeated runs (c.f . Table V). Detailed results are listed in Tables VII-XVI in the Appendix.

V. Conclusions

In this paper proposed an algorithm to solve the problem of work scheduling of employees to minimize the makespan of the overall work of the team. It is assumed that a given task has a different execution time for a given team member and may be executed only by one team member. The proposed approach allows us

1) to model team members' skills in executing certain tasks – setting up the execution time of a certain task by a particular team member for being much longer than the execution time for the other members one can model the situation that some members are not capable of executing this task,
2) synchronization of meetings for the whole team or for a subset of members of a team,
3) model individual working hours in any number of time slots,
4) model that some team members work only part-time,
5) minimize the execution time of a set of tasks.

The latter feature is achieved by reducing the problem to scheduling on unrelated parallel machines $R||C_{max}$. Therefore, any method to solve this problem may be used for schedule calculation. In this paper we exploited for this purpose Genetic Algorithms and Tabu Search as well as compared results with the exact branch and bound method. The obtained results show that starting from a fully random assignment of tasks to team members the schedule can be optimized even up to about 20% of the initial makespan when looking at the wall time of the execution on instances that are difficult to MILP methods. To our knowledge, the presented method can fully model working time organization in corporations allowing for its makespan optimization for whole teams.

Acknowledgment

The authors would like to thank Mr. Tomasz Gniazdowski who helped to implement an optimization framework. This work was partially financed from the grant POIR.01.01.01-00-1831/20 entitled "Development of the NEULA Personal

TABLE III: The mean differences in total task execution time between the initial population and results from Grouping Genetic Algorithm [33] using various crossover operations, i.e., `min`, `MSXF`, `2 points`, `OBX` and mutation operators such as `download` and `usual random` mutation. The initial populations were generated using `min` and `random` methods. These results encompass all instances.

	min								random							
	usual mutation				download mutation				usual mutation				download mutation			
Inst	min	MSXF	2p	OBX	min	MSXF	2p	OBX	min	MSXF	2p	OBX	min	MSXF	2p	OBX
1	0,00	0,00	0,00	0,00	6,58	3,36	6,53	0,06	3,92	0,70	1,58	1,00	3,89	4,00	6,37	3,58
2	0,01	0,00	0,00	0,00	0,17	0,43	0,07	0,04	2,90	1,44	2,34	1,81	4,40	3,46	3,72	4,17
3	0,00	0,00	0,00	0,00	0,09	0,19	0,05	0,05	2,69	1,47	1,14	0,70	3,63	3,54	3,18	3,90
4	0,00	0,00	0,00	0,00	0,09	0,03	0,07	0,06	1,73	1,04	0,94	1,44	3,77	4,09	2,97	4,32
5	0,00	0,00	0,00	0,00	0,18	0,07	0,53	0,08	1,89	1,52	0,17	2,34	4,26	3,39	3,79	3,52
6	0,00	0,00	0,00	0,00	0,23	0,09	0,09	0,16	3,38	-0,06	0,49	0,55	3,14	2,97	3,65	3,49
7	0,04	0,00	0,00	0,00	0,27	0,07	0,10	0,05	18,94	7,60	7,01	-2,64	20,63	19,59	19,70	16,16
8	0,00	0,00	0,00	0,00	0,13	3,30	3,63	0,01	2,60	0,15	1,80	1,48	19,86	17,11	7,12	9,27
9	0,00	0,00	0,00	0,00	0,13	0,06	0,02	0,09	16,50	4,90	4,53	4,72	20,39	19,25	15,83	19,79
10	0,00	0,37	0,00	0,00	0,23	0,11	0,05	0,03	2,56	1,78	2,14	1,51	4,35	3,86	2,13	2,68

TABLE IV: The maximum differences in total task execution time between the initial population and results from Grouping Genetic Algorithm [33] using various crossover operations, i.e., `min`, `MSXF`, `2 points`, `OBX` and mutation operators such as `download` and `usual random` mutation. The initial populations were generated using `min` and `random` methods. These results encompass all instances.

	min								random							
	usual mutation				download mutation				usual mutation				download mutation			
Inst	min	MSXF	2p	OBX	min	MSXF	2p	OBX	min	MSXF	2p	OBX	min	MSXF	2p	OBX
1	0,00	0,00	0,00	0,00	16,40	16,40	16,45	0,15	4,70	2,17	2,82	2,15	4,47	4,87	19,83	4,70
2	0,07	0,00	0,00	0,00	0,27	1,83	0,23	0,15	5,10	3,67	3,87	2,18	5,63	5,23	5,70	4,75
3	0,00	0,00	0,00	0,00	0,13	0,45	0,27	0,25	4,60	3,13	3,07	1,08	4,85	4,92	4,35	4,63
4	0,00	0,00	0,00	0,00	0,20	0,08	0,20	0,15	3,08	2,73	1,92	3,88	5,12	5,17	5,32	5,40
5	0,00	0,00	0,00	0,00	0,28	0,18	1,93	0,25	4,48	3,28	2,17	4,50	5,90	5,08	4,52	5,38
6	0,00	0,00	0,00	0,00	0,30	0,18	0,32	0,28	5,02	1,10	2,58	2,07	3,68	4,13	4,55	4,68
7	0,18	0,00	0,00	0,00	0,45	0,17	0,27	0,12	20,78	18,47	17,97	1,13	21,55	20,68	20,60	20,63
8	0,00	0,00	0,00	0,00	0,22	16,18	18,10	0,05	4,27	2,82	3,85	2,67	21,53	21,15	18,17	19,08
9	0,00	0,00	0,00	0,00	0,25	0,17	0,10	0,20	21,35	19,63	18,80	18,68	21,32	20,68	20,08	20,80
10	0,00	1,83	0,00	0,00	0,33	0,32	0,10	0,08	4,68	3,72	3,77	2,82	5,33	4,87	3,88	3,52

TABLE V: The maximal and mean differences in total task execution time between the Tabu Search and Solution using Hybrid Tabu Search algorithm from [34] using various operators for generating neighborhood i.e. `NH1`, `NH2`, `NH3` – see Section III-C. The results encompass all instances.

	maximal			mean		
Inst	`NH1`	`NH2`	`NH3`	`NH1`	`NH2`	`NH3`
1	17,13	17,07	18,12	13,72	13,72	13,91
2	1,05	2,48	1,00	0,84	1,12	0,85
3	1,22	1,20	1,03	0,99	0,97	0,75
4	2,05	2,70	0,90	1,06	1,24	0,81
5	1,00	1,87	1,80	0,86	1,04	1,03
6	2,00	1,92	2,08	1,15	0,97	1,10
7	18,72	0,97	0,82	4,35	0,78	0,70
8	17,80	17,12	18,78	17,06	16,74	17,30
9	1,03	18,17	18,85	0,93	4,27	4,42
10	2,65	1,03	2,12	1,91	0,87	1,13

TABLE VI: Solution obtained by the exact algorithm [35]

Instance No.	1	2	3	4	5
$C_{\max}$	61,52	79,23	62,77	62,03	79,10
Instance No.	6	7	8	9	10
$C_{\max}$	62,38	58,40	61,33	58,85	62,20

Assistant with artificial intelligence enabling autonomous task definition and user efficiency Management based on deep learning of neural networks".

REFERENCES

[1] S. Martello, F. Soumis, and P. Toth, "Exact and approximation algorithms for makespan minimization on unrelated parallel machines," *Discrete applied mathematics*, vol. 75, no. 2, pp. 169–188, 1997.

[2] N. S. Dey and T. Gunasekhar, "A comprehensive survey of load balancing strategies using hadoop queue scheduling and virtual machine migration," *IEEE Access*, vol. 7, pp. 92 259–92 284, 2019. [Online]. Available: doi:10.1109/ACCESS.2019.2927076

[3] M. S. Qureshi, M. B. Qureshi, M. Fayaz, W. K. Mashwani, S. B. Belhaouari, S. Hassan, and A. Shah, "A comparative analysis of resource allocation schemes for real-time services in high-performance computing systems," *International Journal of Distributed Sensor Networks*, vol. 16, no. 8, p. 1550147720932750, 2020. [Online]. Available: doi:10.1177/1550147720932750

[4] A. Keivani, F. Ghayoor, and J.-R. Tapamo, "A review of recent methods of task scheduling in cloud computing," in *2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON)*, 2018, pp. 104–109. [Online]. Available: doi:10.1109/MELCON.2018.8379076

[5] H. Hussain, S. U. R. Malik, A. Hameed, S. U. Khan, G. Bickler, N. Min-Allah, M. B. Qureshi, L. Zhang, W. Yongji, N. Ghani, J. Kolodziej, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, P. Bouvry, H. Li, L. Wang, D. Chen, and A. Rayes, "A survey on resource allocation in high performance distributed computing systems," *Parallel Computing*, vol. 39, no. 11, pp. 709–736, 2013. [Online]. Available: doi:10.1016/j.parco.2013.09.009

[6] R. Zabolotnyi, P. Leitner, and S. Dustdar, "Profiling-based task scheduling for factory-worker applications in infrastructure-as-a-service clouds," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2014, pp. 119–126. [Online]. Available: doi:10.1109/SEAA.2014.42

[7] V. C. Kalempa, L. Piardi, M. Limeira, and A. S. de Oliveira, "Multi-robot preemptive task scheduling with fault recovery: A novel approach to automatic logistics of smart factories," *Sensors*, vol. 21, no. 19, 2021. [Online]. Available: doi:10.3390/s21196536

[8] O. Berman, R. C. Larson, and E. Pinker, "Scheduling workforce and workflow in a high volume factory," *Management Science*, vol. 43, no. 2, pp. 158–172, 1997.

[9] S. Hasgül, I. Saricicek, M. Ozkan, and O. Parlaktuna, "Project-oriented task scheduling for mobile robot team," *Journal of Intelligent Manufacturing*, vol. 20, pp. 151–158, 2009.

[10] B. Fu, W. Smith, D. M. Rizzo, M. Castanier, M. Ghaffari, and K. Barton, "Robust task scheduling for heterogeneous robot teams under capability uncertainty," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1087–1105, 2022.

[11] H. Fei, C. Combes, N. Meskens, and C. Chu, "Endoscopies scheduling problem: A case study," *IFAC Proceedings Volumes*, vol. 39, no. 3, pp. 635–640, 2006, 12th IFAC Symposium on Information Control Problems in Manufacturing. [Online]. Available: doi:10.3182/20060517-3-FR-2903.00323

[12] J. Pempera and C. Smutnicki, "Open shop cyclic scheduling," *European Journal of Operational Research*, vol. 269, no. 2, pp. 773–781, 2018. [Online]. Available: doi:10.1016/j.ejor.2018.02.021

[13] T. Gonzalez and S. Sahni, "Open shop scheduling to minimize finish time," *J. ACM*, vol. 23, no. 4, p. 665–679, oct 1976. [Online]. Available: doi:10.1145/321978.321985

[14] S. H. Jang, T. Y. Kim, J. K. Kim, and J. S. Lee, "The study of genetic algorithm-based task scheduling for cloud computing," *International Journal of Control and Automation*, vol. 5, no. 4, pp. 157–162, 2012.

[15] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem," *Journal of Parallel and Distributed Computing*, vol. 70, no. 1, pp. 13–22, 2010.

[16] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural computing and applications*, vol. 33, pp. 13 075–13 088, 2021.

[17] A. Awad, N. El-Hefnawy, and H. Abdel_kader, "Enhanced particle swarm optimization for task scheduling in cloud computing environments," *Procedia Computer Science*, vol. 65, pp. 920–929, 2015.

[18] H. Izakian, B. T. Ladani, A. Abraham, V. Snasel *et al.*, "A discrete particle swarm optimization approach for grid job scheduling," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 9, pp. 1–15, 2010.

[19] T. Chen, B. Zhang, X. Hao, and Y. Dai, "Task scheduling in grid based on particle swarm optimization," in *2006 Fifth International Symposium on Parallel and Distributed Computing*. IEEE, 2006, pp. 238–245.

[20] J. P. B. Mapetu, Z. Chen, and L. Kong, "Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing," *Applied Intelligence*, vol. 49, pp. 3308–3330, 2019.

[21] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2020.

[22] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," in *2013 8th international conference on computer engineering & systems (ICCES)*. IEEE, 2013, pp. 64–69.

[23] M. R. Mohamed and M. H. Awadalla, "Hybrid algorithm for multiprocessor task scheduling," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 3, p. 79, 2011.

[24] M. Fahmy, "A fuzzy algorithm for scheduling non-periodic jobs on soft real-time single processor system," *Ain Shams Engineering Journal*, vol. 1, no. 1, pp. 31–38, 2010. [Online]. Available: doi:10.1016/j.asej.2010.09.004

[25] X. Kong, C. Lin, Y. Jiang, W. Yan, and X. Chu, "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction," *Journal of network and Computer Applications*, vol. 34, no. 4, pp. 1068–1077, 2011.

[26] N. Mansouri and M. M. Javidi, "Cost-based job scheduling strategy in cloud computing environments," *Distributed and Parallel Databases*, vol. 38, pp. 365–400, 2020.

[27] M. M. Syslo, N. Deo, and J. S. Kowalik, *Algorytmy optymalizacji dyskretnej: z programami w jezyku Pascal ang: (Algorithms for discrete optimization: with programs in Pascal)*. Wydawnictwo Naukowe PWN, 1999.

[28] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of computational and applied mathematics*, vol. 124, no. 1-2, pp. 281–302, 2000.

[29] Y. Guo, A. Lim, B. Rodrigues, and L. Yang, "Minimizing the makespan for unrelated parallel machines," *International Journal on Artificial Intelligence Tools*, vol. 16, no. 03, pp. 399–415, 2007.

[30] K. Jansen and L. Porkolab, "Improved approximation schemes for scheduling unrelated parallel machines," in *Proceedings of the thirty-first annual ACM Symposium on Theory of Computing*, 1999, pp. 408–417.

[31] S. Balin, "Non-identical parallel machine scheduling using genetic algorithm," *Expert Systems with Applications*, vol. 38, p. 6814–6821, 2011.

[32] Z. Michalewicz, "Genetic algorithms + data structures = evolution programs," 1996.

[33] O. Ramos-Figueroa, M. Quiroz-Castellanos, E. Mezura-Montes, and N. Cruz-Ramirez, "An experimental study of grouping mutation operators for the unrelated parallel-machine scheduling problem," *Mathematical and Computational Applications*, vol. 28, no. 1, 2023. [Online]. Available: doi:10.3390/mca28010006

[34] V. Sels and A. M. D. M. V. José Coelho, "Hybrid tabu search and a truncated branch-and-bound for the unrelated parallel machine scheduling problem," pp. 107–117, 2015.

[35] Åblad Edvin, S. Ann-Brith, and D. Spensieri, "Exact makespan minimization of unrelated parallel machines," 2021.

[36] T. Yamada and R. Nakano, "Scheduling by genetic local search with multi-step crossover," 1996. [Online]. Available: doi:10.1007/3-540-61723-X_1059

[37] D. Goldberg, "Algorytmy genetyczne i ich zastosowania (in polish) [genetic algorithms and their applications]," *WNT, Warsaw*, 1995.

[38] A. J. Umbarkar and P. D. Sheth, "Crossover operators in genetic algorithms: a review." *ICTACT journal on soft computing*, vol. 6, no. 1, 2015.

[39] E. Åblad, D. Spensieri, R. Bohlin, and J. S. Carlson, "Intersection-free geometrical partitioning of multirobot stations for cycle time optimization," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 842–851, 2017.

APPENDIX

The results presented in Tables XVII-XXVI are for the genetic algorithm and in Tables VII-XVI for the tabu-search algorithm. In the tables for the genetic algorithm in each cell there are two numbers. The first is the minimum value within the initial population and for the tabu search is the median of the initial population generated by `min` operator from the genetic algorithm. The median value formed a better starting point, giving a better solution. The second number is the achieved minimum for a chosen setup. The results present the hours count starting from the first task execution. It is a wall time, since the time outside work is also added to the objective function presented. In all the tables in the Appendix, the bold results represent the best results in each column, from five runs. As one can see for the grouping generic algorithm, all the bold results have been achieved when the population initialization is performed with the `min` operator. For the branch and bound algorithm, the final solution is presented in Table VI.

TABLE VII: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neighborhood, i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 1.

	nh1	nh2	nh3
1	79,72/79,03	79,81/62,83	79,93/62,95
2	79,83/62,70	79,78/62,85	81,43/81,00
3	79,68/62,90	79,87/62,80	79,88/**62,72**
4	79,73/**62,65**	79,80/79,17	81,05/62,93
5	79,89/62,88	79,73/**62,75**	79,73/62,90

TABLE VIII: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neghbourhood, i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 2.

	nh1	nh2	nh3
1	81,50/80,63	83,28/80,80	81,48/80,82
2	81,47/80,80	81,48/81,02	81,52/80,73
3	81,42/80,71	81,47/80,67	81,45/**80,53**
4	81,52/80,60	81,55/**80,56**	81,58/80,68
5	81,53/**80,48**	81,53/80,65	81,58/80,58

TABLE IX: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neghbourhood, i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 3.

	nh1	nh2	nh3
1	81,07/**79,85**	81,10/80,28	80,95/80,30
2	81,08/80,13	80,98/80,02	81,10/80,33
3	81,07/80,03	81,10/80,08	81,13/**80,10**
4	80,95/80,08	81,10/80,27	80,98/80,48
5	80,88/80,02	81,12/**79,92**	81,05/80,23

TABLE X: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neghbourhood, i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 4.

	nh1	nh2	nh3
1	80,23/79,50	80,35/79,43	80,18/**79,45**
2	81,22/**79,17**	80,27/79,42	80,32/79,47
3	80,22/79,35	80,27/79,48	80,27/79,37
4	82,28/81,80	80,43/79,48	80,35/79,57
5	80,42/79,27	82,05/**79,35**	80,32/79,53

TABLE XI: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neghbourhood, i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 5.

	nh1	nh2	nh3
1	81,45/80,62	81,37/80,60	81,35/**80,33**
2	81,52/80,67	81,37/**80,37**	81,38/80,45
3	81,43/**80,43**	81,40/80,63	81,28/80,65
4	81,37/80,47	81,40/80,58	81,52/80,77
5	81,43/80,72	82,40/80,53	82,38/80,58

TABLE XII: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neghbourhood, i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 6.

	nh1	nh2	nh3
1	81,58/**79,58**	80,58/79,78	80,48/79,78
2	80,50/79,60	81,62/**79,70**	80,63/79,78
3	80,50/79,60	80,48/79,88	81,82/79,73
4	80,62/79,62	80,55/79,83	80,55/**79,57**
5	80,55/79,60	80,60/79,78	80,53/79,63

TABLE XIII: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neghbourhood, i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 7.

	nh1	nh2	nh3
1	61,67/60,82	61,70/60,77	61,65/61,10
2	61,58/60,85	61,65/61,02	61,53/**60,72**
3	61,58/60,97	61,58/60,93	61,63/60,87
4	61,65/60,82	61,67/**60,70**	61,72/61,03
5	79,50/**60,78**	61,53/60,83	61,72/61,02

TABLE XIV: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neghbourhood i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 8.

	nh1	nh2	nh3
1	79,52/**62,55**	79,47/62,92	81,40/62,62
2	79,50/62,67	79,45/62,85	79,55/62,62
3	79,43/62,63	79,47/62,78	79,47/62,63
4	79,60/62,68	79,60/62,83	79,42/**62,48**
5	80,48/62,68	79,65/**62,53**	79,53/62,53

TABLE XV: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neghbourhood i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 9.

	nh1	nh2	nh3
1	62,25/61,33	62,23/61,43	80,38/61,53
2	62,33/61,37	62,25/61,42	62,22/61,48
3	62,32/61,53	80,35/79,78	62,30/61,47
4	62,22/**61,18**	79,53/61,35	62,20/**61,33**
5	62,28/61,35	62,22/**61,18**	62,25/61,42

TABLE XVI: Solution using the Hybrid Tabu Search algorithm from [34] using various operators for generating the neghbourhood, i.e. `nh1`, `nh2`, `nh3` – see Section III-C. Results show 5 runs for instance no. 10.

	nh1	nh2	nh3
1	82,07/79,45	80,57/79,70	80,40/79,57
2	82,27/79,62	80,47/79,62	80,43/79,65
3	81,85/79,53	80,45/**79,55**	81,58/**79,47**
4	80,38/**79,31**	80,63/79,60	80,52/79,58
5	80,40/79,50	80,55/79,85	80,53/79,55

TABLE XVII: Solution using the Grouping Genetic algorithm from [33] using various crossover operators, i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 1.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	79,08/79,08	79,20/79,20	79,18/79,18	79,12/**79,12**	79,32/79,05	79,27/79,15	79,18/79,15	79,17/79,17
	random	82,30/79,83	84,98/82,82	83,58/82,58	83,02/80,93	82,80/79,12	83,90/79,07	83,75/81,58	82,05/79,35
2	min	79,13/79,13	79,30/79,30	79,28/79,28	79,12/79,12	79,17/79,15	79,22/79,08	79,33/62,88	79,15/79,13
	random	84,15/79,60	82,20/81,90	83,88/82,62	84,85/82,70	82,32/79,08	82,80/79,05	82,28/79,43	82,37/79,27
3	min	79,07/**79,07**	79,18/79,18	79,12/79,12	79,48/79,48	79,32/79,03	79,22/79,08	79,08/62,90	79,18/79,03
	random	84,18/79,48	82,37/81,78	84,98/82,17	81,63/82,82	83,55/79,20	83,33/79,37	82,53/79,35	83,93/79,23
4	min	79,33/79,33	79,20/79,20	79,22/79,22	79,27/79,27	79,40/**63,00**	79,30/**62,90**	79,10/79,10	79,02/**79,02**
	random	83,27/79,45	82,00/82,17	82,88/82,53	82,78/81,53	83,67/79,20	81,70/79,13	82,80/62,97	82,97/79,32
5	min	79,22/79,22	79,13/**79,13**	79,03/**79,03**	79,22/79,22	79,13/62,92	79,05/79,05	62,85/**62,85**	79,23/79,08
	random	83,60/79,52	82,93/82,30	82,95/80,48	83,07/82,38	82,78/79,07	84,12/79,25	83,20/79,37	82,97/79,23

TABLE XVIII: Solution using the Grouping Genetic algorithm from [33] using various crossover operators i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 2.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	80,93/80,87	80,70/**80,70**	80,88/80,88	80,85/80,85	80,80/80,75	80,90/80,87	80,82/80,77	80,83/80,78
	random	83,92/81,30	86,30/82,63	85,73/81,87	85,52/84,15	85,08/80,83	84,52/81,07	86,57/80,87	85,38/81,12
2	min	80,87/80,87	80,90/80,90	80,75/80,75	80,82/**80,82**	80,88/80,63	80,88/80,83	80,82/80,82	80,82/80,82
	random	84,45/81,43	83,20/84,13	82,83/84,20	86,40/84,22	85,10/80,68	84,93/80,98	85,82/80,92	85,18/81,20
3	min	80,63/**80,63**	80,90/80,90	81,07/81,07	80,92/80,92	80,83/80,78	80,78/**80,73**	80,88/**80,65**	80,97/80,97
	random	84,33/81,37	84,90/83,62	85,50/82,43	83,87/82,22	84,63/80,85	86,20/80,97	83,88/81,15	85,28/80,98
4	min	80,82/80,82	80,90/80,90	80,92/80,92	80,93/80,93	80,90/80,63	82,60/80,77	80,82/80,82	80,70/80,70
	random	86,57/81,47	85,60/83,13	86,67/84,17	84,42/82,70	86,58/80,95	83,73/80,97	83,58/81,08	85,88/81,13
5	min	80,75/80,75	81,03/81,03	80,73/**80,73**	80,98/80,98	80,77/**80,53**	80,93/80,77	80,90/80,82	80,85/**80,70**
	random	83,98/83,20	83,02/82,30	86,02/82,37	84,78/82,63	84,75/80,83	82,83/80,93	84,00/81,22	84,55/81,00

TABLE XIX: Solution using the Grouping Genetic algorithm from [33] using various crossover operators, i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 3.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	80,55/80,55	80,50/80,50	80,53/80,53	80,33/80,33	80,47/80,33	80,58/**80,13**	80,47/80,47	80,38/80,38
	random	83,62/80,78	83,35/82,50	85,33/82,27	83,40/83,00	82,97/80,30	85,07/80,33	85,05/80,70	84,45/80,60
2	min	80,42/80,42	80,28/**80,28**	80,45/**80,45**	80,35/80,35	80,37/80,35	80,42/80,42	80,68/80,42	80,40/80,40
	random	84,38/80,75	85,78/82,65	82,80/82,30	83,65/83,57	85,28/80,43	82,50/80,43	85,57/82,70	84,27/80,53
3	min	80,38/80,38	80,57/80,57	80,55/80,55	80,50/80,50	80,28/**80,23**	80,37/80,37	80,07/**80,07**	80,38/80,38
	random	83,98/82,48	83,50/81,45	85,18/83,95	84,95/83,88	84,43/80,38	83,72/80,37	84,00/80,38	84,40/80,48
4	min	80,30/**80,30**	80,32/80,32	80,50/80,50	80,52/80,52	80,52/80,38	80,55/80,25	80,32/80,32	80,37/80,37
	random	83,73/82,83	82,32/82,28	83,13/83,62	83,83/82,95	83,78/80,35	82,97/80,32	84,00/82,43	83,92/80,55
5	min	80,50/80,50	80,50/80,50	80,57/80,57	80,22/**80,22**	80,50/80,38	80,43/80,25	80,37/80,37	80,60/**80,35**
	random	85,63/81,03	83,05/81,77	83,27/81,87	83,75/82,67	83,42/80,28	85,57/80,65	84,12/80,62	85,53/80,90

TABLE XX: Solution using the Grouping Genetic algorithm from [33] using various crossover operators, i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 4.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	79,63/79,63	79,77/79,77	79,77/79,77	79,62/79,62	79,63/79,63	79,72/79,63	79,53/79,53	79,68/**79,53**
	random	83,25/80,17	84,07/81,90	84,02/82,55	82,35/83,15	82,12/79,43	83,67/79,78	81,48/79,78	84,43/79,75
2	min	79,63/79,63	79,63/**79,63**	79,80/79,80	79,78/79,78	79,72/**79,52**	79,67/79,65	79,85/79,65	79,67/79,58
	random	81,58/80,27	82,70/83,07	83,47/83,02	82,13/82,87	84,63/79,52	83,10/79,58	83,78/79,97	85,18/79,78
3	min	79,70/79,70	79,78/79,78	79,62/79,62	79,77/79,77	79,60/79,58	79,58/79,58	79,58/79,58	79,57/79,57
	random	82,83/80,08	81,58/80,97	81,10/81,10	84,00/82,87	83,15/79,60	84,80/79,63	81,88/82,00	83,80/79,62
4	min	79,62/**79,62**	79,68/79,68	79,85/79,85	79,55/**79,55**	79,75/79,62	79,60/79,55	79,63/**79,48**	79,62/79,62
	random	83,00/82,10	83,58/80,85	84,43/82,52	85,37/81,48	83,02/79,45	85,22/80,13	83,85/79,72	84,35/79,60
5	min	79,72/79,72	79,73/79,73	79,58/**79,58**	79,67/79,67	79,65/79,55	79,40/**79,40**	79,78/79,78	79,68/79,60
	random	80,78/80,18	82,35/82,28	81,80/80,92	84,52/80,80	83,63/79,70	84,62/81,80	85,28/79,97	82,38/79,78

TABLE XXI: Solution using the Grouping Genetic algorithm from [33] using various crossover operators, i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 5.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	80,67/**80,67**	80,70/**80,70**	80,82/80,82	80,68/**80,68**	80,83/80,63	80,75/**80,57**	82,73/80,80	80,47/**80,47**
	random	83,80/81,38	86,42/83,13	83,77/84,63	84,18/84,03	84,17/80,63	83,15/80,85	85,20/80,70	84,10/83,07
2	min	80,72/80,72	80,82/80,82	80,65/**80,65**	80,73/80,73	80,75/80,55	80,68/80,68	80,80/80,53	80,73/80,67
	random	83,28/82,85	86,02/84,02	82,70/83,80	86,47/82,10	83,47/80,60	85,88/80,80	85,32/80,80	83,58/81,02
3	min	80,72/80,72	80,92/80,92	80,68/80,68	80,72/80,72	80,70/**80,47**	80,77/80,67	80,90/**80,53**	80,62/80,55
	random	84,50/83,25	83,52/84,13	86,00/83,83	85,40/82,88	85,52/80,72	84,10/80,62	83,30/80,73	85,38/80,92
4	min	80,82/80,82	80,75/80,75	80,83/80,83	80,85/80,85	80,82/80,53	80,62/80,62	80,68/80,68	80,80/80,80
	random	84,13/83,27	84,17/83,80	82,98/82,40	82,35/82,20	86,60/80,70	84,32/80,93	84,60/80,92	86,25/80,87
5	min	80,67/80,67	80,78/80,78	80,88/80,88	80,87/80,87	80,57/80,57	80,83/80,75	80,78/80,70	80,80/80,55
	random	85,65/81,17	84,85/82,27	84,17/84,12	86,43/81,93	84,93/80,72	85,20/82,52	84,55/80,87	85,17/81,02

TABLE XXII: Solution using the Grouping Genetic algorithm from [33] using various crossover operators, i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 6.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	79,90/79,90	80,07/80,07	79,82/79,82	80,00/80,00	79,92/79,75	79,98/79,80	79,92/**79,60**	80,00/**79,75**
	random	82,50/80,32	81,97/83,18	83,25/83,07	85,25/83,18	83,42/79,77	81,53/80,07	83,18/80,03	84,98/80,30
2	min	79,93/79,93	80,02/80,02	79,90/79,90	79,82/**79,82**	79,90/**79,72**	79,97/**79,78**	79,83/79,83	80,08/79,80
	random	82,73/80,52	83,37/82,90	81,93/82,38	81,97/82,35	82,78/79,82	83,95/79,82	84,37/79,82	82,32/80,20
3	min	79,85/**79,85**	80,00/80,00	79,90/79,90	79,93/79,93	80,02/79,72	79,88/79,88	80,00/79,90	79,90/79,90
	random	82,90/80,40	84,38/83,28	84,50/83,55	82,50/83,13	82,60/79,90	82,13/80,10	82,43/80,00	83,00/80,12
4	min	79,98/79,98	79,98/**79,98**	79,78/**79,78**	80,03/80,03	79,97/79,75	79,90/79,90	79,88/79,83	79,97/79,97
	random	85,52/80,52	81,68/81,32	85,00/82,42	84,30/82,62	83,63/79,95	83,15/79,98	84,77/80,25	83,62/79,98
5	min	79,85/79,85	80,05/80,05	79,98/79,98	79,83/79,83	80,00/79,70	80,00/79,90	79,82/79,82	80,05/79,78
	random	85,18/80,17	82,35/83,38	82,65/83,48	82,78/82,78	82,55/79,83	83,98/79,93	83,82/80,23	86,08/81,93

TABLE XXIII: Solution using the Grouping Genetic algorithm from [33] using various crossover operators, i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 7.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	61,22/61,22	61,25/61,25	61,18/61,18	61,03/61,03	61,08/60,82	61,03/61,03	61,15/61,08	61,08/61,05
	random	82,30/61,52	82,80/80,80	79,72/79,88	81,50/80,40	81,88/61,02	81,90/61,22	79,30/61,12	80,87/61,25
2	min	61,12/61,12	61,15/61,15	61,05/61,05	61,02/**61,02**	61,17/60,90	61,02/61,02	61,32/61,13	61,05/61,05
	random	79,92/61,62	79,70/62,45	79,83/80,15	62,92/79,92	80,82/61,00	80,47/61,27	81,90/61,30	81,92/61,28
3	min	61,20/**61,02**	61,08/61,08	61,02/61,02	61,18/61,18	61,05/60,93	61,03/**60,93**	61,07/61,07	60,98/60,98
	random	80,53/61,53	80,38/80,10	62,75/62,58	81,20/80,25	82,00/61,03	79,43/61,33	81,42/61,18	82,07/61,78
4	min	61,12/61,12	61,22/61,22	60,95/60,95	61,05/61,05	61,07/60,83	61,08/60,98	61,03/**61,03**	61,15/61,03
	random	79,58/61,43	80,48/80,48	79,72/62,33	81,25/80,12	80,97/61,02	80,50/61,17	81,45/61,25	79,87/61,33
5	min	61,25/61,25	61,02/**61,02**	61,08/61,08	60,87/60,87	61,18/**60,73**	61,13/60,97	61,27/61,00	61,05/**60,97**
	random	79,83/61,37	81,45/62,98	80,17/62,20	81,40/80,80	82,58/61,03	81,77/61,15	80,58/61,28	62,87/61,15

TABLE XXIV: Solution using the Grouping Genetic algorithm from [33] using various crossover operators, i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 8.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	62,82/62,82	62,95/62,95	62,97/62,97	62,75/**62,75**	62,67/**62,58**	62,88/62,77	62,72/**62,72**	62,88/62,88
	random	81,48/81,30	81,40/82,02	83,90/80,05	82,93/82,33	84,17/62,63	82,65/62,90	82,15/79,17	80,47/79,08
2	min	62,67/**62,67**	62,88/62,88	62,88/62,88	63,00/63,00	62,97/62,77	63,00/62,88	80,92/62,82	62,50/**62,50**
	random	83,12/79,17	84,28/81,47	83,35/80,50	83,73/81,23	82,17/62,93	81,57/62,90	83,73/79,03	81,42/62,98
3	min	62,78/62,78	62,77/**62,77**	62,75/**62,75**	79,02/79,02	62,85/62,70	62,90/62,88	62,73/62,73	62,87/62,85
	random	82,93/81,07	80,30/82,20	81,72/82,08	84,65/81,98	83,48/62,88	84,07/62,92	84,27/79,03	82,28/79,05
4	min	62,98/62,98	62,88/62,88	79,03/79,03	79,07/79,07	62,97/62,75	79,03/62,85	62,90/62,90	62,85/62,85
	random	82,03/79,30	82,85/82,10	82,83/81,80	83,65/82,42	81,92/62,87	83,93/62,88	83,58/79,07	83,40/79,20
5	min	62,78/62,78	62,83/62,83	79,05/79,05	62,85/62,85	62,72/62,72	62,78/**62,72**	62,87/62,83	62,85/62,80
	random	83,53/79,27	82,08/82,38	82,20/80,58	81,68/81,30	81,62/62,72	84,07/79,13	81,08/62,92	82,07/62,98

TABLE XXV: Solution using the Grouping Genetic algorithm from [33] using various crossover operators, i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 9.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	61,60/61,60	61,77/61,77	61,65/61,65	61,68/61,68	61,53/61,47	61,68/61,57	61,65/61,65	61,68/61,68
	random	81,20/79,95	80,85/79,87	82,97/80,87	81,88/79,20	82,87/61,55	82,42/61,88	79,78/61,78	82,20/61,85
2	min	61,67/61,67	61,60/**61,60**	61,62/61,62	61,68/61,68	61,67/**61,42**	61,57/61,57	61,58/61,58	61,65/61,58
	random	83,47/62,12	82,23/80,93	79,90/81,13	80,10/80,97	82,03/61,72	79,60/61,90	80,18/61,78	81,72/61,77
3	min	61,35/**61,35**	61,70/61,70	61,72/61,72	61,72/61,72	61,62/61,45	61,53/61,53	61,70/61,60	61,53/61,53
	random	80,75/62,12	81,82/62,18	79,55/79,20	82,23/79,12	81,67/61,50	82,33/61,65	81,88/61,95	82,10/61,80
4	min	61,58/61,58	61,70/61,70	61,53/**61,53**	61,50/**61,50**	61,75/61,60	61,75/61,58	61,72/61,72	61,67/**61,47**
	random	82,48/62,13	82,00/80,35	82,85/80,20	79,05/79,05	81,52/61,70	79,73/61,77	82,23/79,52	79,67/62,13
5	min	61,70/61,70	61,68/61,68	61,65/61,65	61,63/61,63	61,45/61,45	61,53/**61,52**	61,43/**61,43**	61,73/61,57
	random	83,02/62,12	81,83/80,92	81,35/62,55	81,65/62,97	81,80/61,45	80,93/61,58	81,72/61,63	82,53/61,73

TABLE XXVI: Solution using the Grouping Genetic algorithm from [33] using various crossover operators, i.e. `min`, `MSXF`, `2 points`, `OBX` and mutations operators such as `download`, `min` and initial population creation `random` and `min`. Results show 5 runs for instance no. 10.

		usual mutation				download mutation			
		min	MSXF	2points	OBX	min	MSXF	2points	OBX
1	min	79,77/79,77	81,83/80,00	79,90/79,90	79,73/79,73	79,92/**79,58**	79,78/79,78	79,80/**79,78**	79,82/79,82
	random	83,45/79,97	84,50/83,17	83,60/82,20	85,03/82,22	84,20/79,78	82,88/80,07	82,62/82,28	82,40/80,12
2	min	79,83/79,83	79,90/79,90	79,72/**79,72**	79,67/79,67	79,98/79,73	79,88/79,82	79,90/79,82	79,77/**79,68**
	random	84,83/80,15	84,23/83,32	84,97/81,28	84,02/81,78	84,10/79,82	84,18/79,90	84,13/80,25	83,05/79,85
3	min	79,70/**79,70**	79,90/79,90	79,78/79,78	79,88/79,88	79,88/79,73	79,60/**79,60**	79,90/79,90	79,80/79,80
	random	82,97/82,47	84,30/80,58	83,77/82,07	82,40/82,13	83,27/79,83	84,95/80,08	84,53/82,15	84,83/82,28
4	min	79,97/79,97	79,87/79,87	79,82/79,82	79,55/**79,55**	79,83/79,65	80,03/79,72	79,85/79,78	79,80/79,80
	random	82,50/80,13	85,48/82,93	85,08/81,32	82,03/81,28	85,02/79,68	83,65/80,00	82,52/81,68	81,53/79,70
5	min	80,03/80,03	79,80/**79,80**	79,92/79,92	79,60/79,60	79,90/79,65	79,95/79,80	79,98/79,88	79,85/79,80
	random	82,25/80,48	81,72/81,35	83,03/82,88	84,52/83,02	84,18/79,88	83,57/79,90	85,12/81,92	83,67/80,15