Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

# An evolutionary algorithm for joint bi-criteria location-scheduling problem

**Grzegorz Filcek[a*], Jerzy Józefczyk[a] and Mirosław Ławrynowicz[a]**

[a]*Department of Computer Science and Systems Engineering, Wroclaw University of Science and Technology, Wroclaw, Poland, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland*

**C H R O N I C L E**

*Article history*:
Received July 10 2020
Received in Revised Format November 3 2020
Accepted December 3 2020
Available online December, 7 2020

Keywords:
*ScheLoc problem*
*Multi-criteria optimization*
*Evolutionary algorithm*
*Evacuation*

**A B S T R A C T**

A new case of joint location and scheduling (ScheLoc) problem is considered. It deals with selecting a non-fixed number of locations for identical parallel executors (machines) from a given set of available sites. Simultaneously, a schedule for a set of tasks is sought. For every task, it comprises an executor carrying-out the task and the moment of time when the performance of the task is started. The locations for executors and the schedule are evaluated by two criteria: the sum of task completion times and investment costs incurred when locations for executors are selected and launched. It is justified that the joint optimization problem is strongly NP-hard. In consequence, a heuristic algorithm Alg_BC is proposed, which uses the general scheme of NSGA II provided for the multi-criteria optimization. The performance of Alg_BC is evaluated for small instances by exact solutions determined by the Matlab solver. The sensitivity analysis for bigger instances is also provided, which among others, allows examining the influence of both component criteria on results generated by the evaluated algorithm. A case study dealing with the evacuation of citizen groups from danger zones is provided as an example of the investigated bi-criteria ScheLoc problem. The usefulness of Alg_BC is confirmed as well.



## 1. Introduction

Investigation of topics of complex optimization or decision making problems can be observed in operations research and discrete optimization. In such problems, some interconnected sub-problems can be distinguished. Neglecting the interconnections among sub-problems is a simple paradigm constituting solution algorithms of the mentioned problems frequently met in the initial phase of their investigation. Advances in the methodology and computing potential of solution algorithms facilitate dealing with such problems as a whole, and, as a consequence, it is possible to obtain improved solutions. Let us mention project scheduling-location, replenishment-location, location-routing, production-inventory-location, routing-scheduling as examples of complex problems, where sub-problems are solved jointly, (e.g., Rostami & Bagherpour, 2017; Pasandideh et al., 2018; Prodhon & Prins, 2014; Shahabi et al., 2018; Paraskevopoulos et al., 2017).

This paper's investigations are addressed at another complex problem called location-scheduling. Scheduling of a given set of tasks and location planning of executors (machines) as performers of the tasks are interconnected parts (sub-problems) of a complex problem. It is assumed that a finite number of tasks are deployed at given original locations in a planar area. Every task has to be moved from its original location to the position of the corresponding executor site, which is not known in advance. Then, all the tasks moved to the same executor are scheduled.

Consequently, the purpose of the complex problem is to jointly deploy a set of executors in a planar area and schedule a set of tasks on previously located executors such that relevant criteria evaluating both activities are optimized. There are many applications described in the literature where both location and scheduling are required. The unloading of ships by cranes is

* Corresponding author
E-mail: grzegorz.filcek@pwr.edu.pl (G. Filcek)

2021 Growing Science Ltd.
doi: 10.5267/j.ijiec.2020.12.002

the first example (Scholz, 2012). Unloading ships waiting at sea can be interpreted as tasks. Locations for cranes as executors at an embankment need the determination together with the order of ships' service after their reaching the embankment. Kalsch and Drezner (2010) invoke a similar application where one has to find locations for ships on a berth and schedule the loading process of containers. The operation of movable crushers in the mining industry described in Kalsch (2009) is another example. The evacuation of groups of people from danger zones can be indicated as the next example (Heßler, 2017). Each group can be treated as a task performed by a salvage crew being an executor. Locations of the so-called concentration points where groups of people have to gather are the subject of a decision. Simultaneously a schedule of the evacuation is needed. Another example concerns modern computer distributed systems where a given number of customer tasks wait for a computation service provided by selected servers. Such servers have to be previously selected from the given set of available servers, and, then, they are assigned in time to perform the customer tasks (Saldana and Suznjevic, 2017).

### *1.1 Related work*

Such a introduced complex problem referred to as ScheLoc has been considered for the first time in Hennes and Hamacher (2002). Then, it has been discussed and developed in many works. Many particular cases have been taking into consideration depending on the kind of location area for executors, a type and number of executors, the criterion for evaluating a schedule of tasks as well as used solution algorithms. A summary of the considered cases is provided in Table 1, see also Liu et al. (2019). A majority of works deal with a discrete area for the deployment of executors where a finite set of available positions for executors is known and given a priori (e.g., Heßler and Deghdak, 2017; Liu et al., 2019; Ławrynowicz and Józefczyk, 2019). A non-empty subset of this set is to be selected. In many works, a one-element subset is assumed. As a consequence, only the location of a single executor is sought. Two task scheduling criteria, i.e., the makespan $C_{\max}$ and the sum of completion times $\sum C_j$ (Pinedo, 2012), have been noticed in the literature as the evaluation of a task schedule. The criteria serve in the analyzed works as the assessment of ScheLoc as a whole. Other possibly used criteria assume in some works the supporting role for the deployment of executors, (e.g., Kalsch and Drezner, 2010; Ławrynowicz and Józefczyk, 2019). Authors proposed different heuristic algorithms due to a severe time complexity not only of the joint problem but also of its component sub-problems. Let us remember that the task scheduling sub-problem with different release dates is strongly NP-hard for a single executor and criterion $\sum C_j$ (Lenstra et al., 1977). Corresponding sub-problems for at least two executors and criterion $C_{\max}$ are NP-hard (Pinedo, 2012). The other sub-problem is also NP-hard when the deployment of executors is treated as a particular case of the uncapacitated facility location problem (UFLP) or p-median problem (e.g., Krarup and Pruzan, 1983).

As has been mentioned, ScheLoc has been presented for the first time in Hennes and Hamacher (2002), where a straightforward case is discussed. Available positions for a single executor are limited to a graph. An integrated model is provided in which executor locations define release dates for tasks. The polynomial solution algorithm is presented, comprising locational and scheduling parts. Release dates are calculated in the former part based on the distance among positions of tasks and the position of the executor. Then, the scheduling part is solved by the ERD rule.

Two similar works of Elvikis et al. (2007) and Elvikis et al. (2009) focus on the same task scheduling problem $1 \mid r_j(x) \mid C_{\max}$ where $r_j(x)$ is the release date dependent on the executor's position $x$. The locational analysis is substantially extended in comparison with previous works. In particular, the continuous case is considered, and various distance functions are studied. The ERD rule is used as well, which uses the values of release dates calculated based on regions in the available area where the dates do not change. The authors present three polynomial algorithms. One constructive algorithm uses geometric properties. Two other propositions contain linear programming models that are solved by the Matlab solver.

The idea of calculating release dates for task scheduling is continued in Kalsch and Drezner (2010). The sum of completion times of tasks $\sum C_j$ is considered apart from $C_{\max}$ but again for a single executor. The solution algorithm of the scheduling part is based on the ERD rule, which requires equal task processing times when the criterion $\sum C_j$ is minimized. The core investigations are concentrated on the locational analysis for nonconvex cases with the use of the big triangle small triangle approach (Drezner and Suzuki, 2004). The version with many executors has been introduced in Rajabzadeh et al. (2016), where both discrete and continuous planar areas are considered. For the criterion $C_{\max}$, the mathematical models are presented, and the solver GAMS is employed as a solution tool. The authors provide examples of results limited to ten tasks and three executors.

The task scheduling on many parallel identical executors, the makespan $C_{\max}$ as the criterion, and discrete locations for executors are investigated in Heßler and Deghdak (2017), and Heßler (2017). The constructive algorithm has been proposed that is based on the iterations between location and assignment sub-problems. In the first step, the algorithm starts with a single cluster containing all jobs, and, for this cluster, a single optimal machine location is computed. In the second step, a set of jobs is removed from the existing cluster, and a new optimal location is set. The procedure continues until it reaches the number of demanded locations.

**Table 1**
Summary of joint location-scheduling

| Location area | | | Criteria | | | Executors | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| discrete | network | continuous | $C_{\max}$ | $\sum C_j$ | other | single | identical | unrelated | other | References |
| - | + | - | + | - | - | + | - | - | - | Hennes & Hamacher, 2002 |
| - | - | + | + | - | - | + | - | - | - | Elvikis et al., 2009 |
| - | - | + | + | + | - | + | - | - | - | Kalsch & Drezner, 2010 |
| + | - | + | + | - | - | - | + | - | - | Rajabzadeh et al., 2016 |
| + | - | - | + | - | - | - | + | - | - | Hessler & Deghak, 2017 |
| - | - | + | - | + | - | - | + | - | - | Piasecki, 2018; Piasecki & Józefczyk, 2018 |
| + | - | - | - | - | + | | + | - | - | Liu et al., 2019 |
| + | - | - | + | - | - | - | - | + | - | Ławrynowicz & Józefczyk, 2019 |
| + | - | - | - | - | + | - | + | - | - | Current paper |

The Master Thesis of Piasecki (2018) and the following conference paper of Piasecki and Józefczyk (2018) extend the investigations on the minimization of $\sum C_j$ with the development of many parallel identical executors in the continuous area. The evolutionary algorithm solves the resultant strongly NP-hard problem. The results for small instances are compared with optimal outcomes generated by the Matlab solver.

A more advanced complex problem is discussed in Ławrynowicz and Józefczyk (2019). Namely, many unrelated parallel executors should be located in a discrete area, and apart from $C_{\max}$, the additional criterion has been introduced in the form of the sum of release dates for tasks. The joint problem is considered as the composition of two sub-problems, i.e., the location of executors (LE) and the scheduling of tasks (ST). Two approaches are proposed, solved and compared. The first one, referred to as the sequential approach, comprises two steps: solving the LE sub-problem to minimize the sum of release dates and ST to minimize $C_{\max}$ with the release dates calculated based on the previously solved LE sub-problem. The known algorithms have been employed to solve both sub-problems. In the second approach called the systemic approach, both sub-problems are solved jointly to minimize $C_{\max}$. Executor locations are determined to minimize $C_{\max}$ not the sum of release dates, like in the sequential approach. The memetic algorithm based on Tabu Search metaheuristics has been proposed as the solution tool. The conducted experimental comparison confirmed the substantial advantage of the systemic approach in comparison with the sequential one.

It is also worth pointing out works expanding the topic directly connected with ScheLoc. Liu et al. (2019) consider the stochastic task execution times and the expected value of $\sum C_j$ together with the total location costs of executors as two criteria. However, the authors apply a simple scalarization, and, in consequence, a single criterion optimization problem is solved. A genetic algorithm and the sample average approximation algorithm were elaborated for the large and small instances, respectively. In turn, four criteria are taken into consideration in Wesolkowski et al. (2014) while assessing locational and assignment decisions for military applications. The solution algorithm is based on the NSGA II approach for solving the resulting multi-objective optimization problem.

### *1.2 Contribution of Paper*

In comparison with previous works, the contribution of this paper can be explained as follows.

1. First of all, a new optimization problem is solved. A single criterion has evaluated all variables (decisions) in previous works. The distances and release dates, which have been used in some works for the determination of executor locations, were auxiliary criteria. The makespan $C_{\max}$, or the sum of completion times $\sum C_j$, was the main criterion, and it assessed the schedule of tasks. The quality of the deployment of executors was only indirectly evaluated by $C_{\max}$ or $\sum C_j$ as they were the basis for the calculation of release dates. We propose to consider the criterion which assesses the cost of executor locations apart from the criterion $\sum C_j$ evaluating the schedule of tasks. In consequence, a new bi-criteria joint location-scheduling problem is directly solved. Consequently, we search for a Pareto front of solutions, not for a single solution like, e.g., in Liu et al. (2019), where the simplification via scalarization is applied, and a unique solution is determined.
2. In the face of strong NP-hardness of the optimization problem, the time-efficient heuristic algorithm is proposed. It is based on the NSGA II scheme (Deb, 2001).

3. The usefulness of the algorithm has been experimentally confirmed via comparison with the Matlab solver outcomes or sensitivity analysis for small or big instances, respectively. We propose bespoke performance indices for the evaluation, being the adaptation of known indices, which allows for a better assessment of resulted Pareto fronts.
4. A case study concerning the evaluation has been provided, which illustrates the application potential of the problem.

The remainder of the paper is organized as follows. The mathematical model is provided in Section 2, which is followed by the presentation in Section 3 of the evolutionary algorithm. Section 4 is devoted to the computational experiments, which, first of all, allowed us to evaluate the solutions determined by the proposed algorithm by the exact solutions given by the Matlab solver. A case study presenting the application of the algorithm is provided in Section 5. Final remarks complete the paper.

## 2. Problem formulation

Let us consider a set $J = \{1,2,...,j,...,n\}$ of *n* tasks and a set $A = \{a_1, a_2, ..., a_j, ..., a_n\}$ of their known locations where the vector $a_j = [a_j^{(1)}, a_j^{(2)}]^{\mathrm{T}}$ is the location of the task *j*. The current task *j* is characterized by execution time $p_j$, ready time $\rho_j$, as well as transportation speed $v_j$, and it needs to be performed by a single executor (machine) selected from a set of identical machines referred hereafter to as executors. Executors' prospective locations should be selected from a set $B = \{b_1, b_2, ..., b_i, ..., b_\mu\}$ of $\mu$ possible locations where $b_i = [b_i^{(1)}, b_i^{(2)}]^{\mathrm{T}}$. We assume that the number *m*, $1 \le m \le \mu$ of employed executors, which is equal to the number of selected locations from set *B*, is not known a priori.

Let us introduce a vector $y = [y_i]^{\mathrm{T}}_{i=\overline{1,\mu}}$, where $y_i = 1(0)$ if the location $b_i$ is selected for the deployment of an executor, taken from the set of identical executors (otherwise). Concurrently, the index *i* denotes an executor deployed at the location $b_i$. First of all, a schedule of tasks is sought, which can be represented by a three-dimensional binary matrix $x = [x_{jik}]_{\substack{j,k=\overline{1,n} \\ i=\overline{1,\mu}}}$ in which current entry $x_{jik}$ is equal to 1(0) if the *j*th task is scheduled on the *i*th executor as the *k*th (otherwise). Performance of task *j* by executor *i* follows its transportation at a distance $d(a_j, b_i)$ between locations $a_j$ and $b_i$ if $y_i = 1$. So, the task *j* is composed of two parts: transportation from the initial location $a_j$ and the essential performance at the selected executor's location $b_i$. The first part needs the time

$$r_j(x,y) = \rho_j + \frac{1}{v_j} \sum_{k=1}^{n} \sum_{i=1}^{\mu} d(a_j, b_i) y_i x_{jik} \tag{1}$$

interpreted as a release date for the task $j$, which has to be performed once by one executor. The time $p_j$ is allotted for the execution of the second part. We employ two conflict criteria for the evaluation of decisions $x$ and $y$. The first one as the sum of completion times of all the tasks evaluates the schedule $x$, but it certainly depends on the locational decisions $y$:

$$q^{(1)}(x,y) = \sum_{i=1}^{\mu} \sum_{k=1}^{n} y_i C_{ik}(x,y) \tag{2}$$

where the auxiliary variable $C_{ik}(x,y)$ stands for the completion time of a job performed by the *i*th executor as the *k*th and depends on decision variables via constraints (6) and (7) (Kooli and Serairi, 2014). The total location (investment) cost of all locations used by executors serves as the second criterion:

$$q^{(2)}(y) = \sum_{i=1}^{\mu} c_i y_i \tag{3}$$

where $c_i$ is the location cost of $b_i$. Obviously, launching fewer locations *m* decreases the investment cost $q^{(2)}(y)$, but, simultaneously, it increases the sum of completion times $q^{(1)}(x,y)$, and vice versa. Therefore, the trade-off must be sought, which leads to the bi-criteria problem. The following constraints imposed on decision variables $x$ and $y$ enable having feasible solutions:

$$\sum_{i=1}^{\mu} \sum_{k=1}^{n} x_{jik} = 1, j = 1,2,...,n, \tag{4}$$

$$\sum_{j=1}^{n} x_{jik} \le y_i, i = 1,2,...,\mu, k = 1,2,...,n, \tag{5}$$

$$C_{ik}(x,y) \ge \sum_{j=1}^{n} (r_j(x,y) + p_j) x_{jik}, \quad i = 1,2,...,\mu, k = 1,2,...,n, \tag{6}$$

$$C_{ik}(x,y) \ge C_{i,k-1}(x,y) + \sum_{j=1}^{n} p_j x_{jik}, \quad i = 1,2,...,\mu, k = 2,3,...,n, \tag{7}$$

$$1 \le \sum_{i=1}^{\mu} y_i \le \mu, \tag{8}$$

$$\sum_{j=1}^{n} (x_{j,i,k+1} - x_{jik}) \ge 0, i = 1,2,...,\mu, k = 1,2,...,n-1, \tag{9}$$

$$y_i, x_{jik} \in \{0,1\}, j,k = 1,2,...,n, i = 1,2,...,\mu. \tag{10}$$

Constraints (4) and (5) ensure that each task is performed on one position of a single executor, and no more than one task can be performed on the launched location of each executor, respectively. Moreover, in a feasible solution, the *k*th task in order on the *i*th executor cannot start before its release date or the completion time of a task which is scheduled on the (*k*-1)th position of the same executor. Constraints (6) and (7) fulfil this condition. The next constraint guarantees that the number of launched locations equal to the number of used executors is between 1 and $\mu$. According to (9), all tasks assigned to an individual executor have to be represented by a single sequence of '1's. It means that any sub-sequence of zeros within the whole sequence can occur. In consequence, the number of equivalent solutions represented by matrix $x$ is limited. Finally, constraint (10) defines the domains for decision variables. In consequence, the following bi-criteria optimization problem, referred to as BC_ScheLoc (Bi-Criteria ScheLoc), is solved. Given $n$, $A$, $B$, $\mu$, $p_j$, $v_j$, $\rho_j$, $c_i$, $j = 1,2,...,n$, $i = 1,2,...,\mu$ find the schedule of tasks $x$ and the locations of executors $y$ minimizing a vector of criteria $q(x,y) = [q^{(1)}(x,y), q^{(2)}(y)]^{\mathrm{T}}$ subject to constraints (4)-(10). Let us point out that the number of used executors $m$ as the indirect result of optimization can be calculated according to the formulae $m = \sum_{i=1}^{\mu} y_i$.

**Preposition:** Problem BC_ScheLoc is NP-hard in a strong sense.

*Proof* Let us consider a special case of BC_ScheLoc when $\mu = m = 1$, and values of $y$ is known and fixed. Then $q^{(2)}(y)$ is constant, and $q^{(1)}(x,y)$ along with constraints (4)-(10) can be simplified to

$$q^{(1)}(\overline{x}) = \sum_{k=1}^{n} \overline{C}_k \tag{11}$$

$$\sum_{k=1}^{n} \overline{x}_{jk} = 1, j = 1,2,...,n, \tag{12}$$

$$\sum_{j=1}^{n} \overline{x}_{jk} \le 1, k = 1,2,...,n, \tag{13}$$

$$\overline{C}_k \ge \sum_{j=1}^{n} (r_j + p_j) \overline{x}_{jk}, k = 1,2,...,n, \tag{14}$$

$$\overline{C}_k \ge \overline{C}_{k-1} + \sum_{j=1}^{n} p_j \overline{x}_{jk}, k = 2,3,...,n, \tag{15}$$

$$\sum_{j=1}^{n} (\overline{x}_{j,k+1} - \overline{x}_{jk}) \ge 0, k = 1,2,...,n-1, \tag{16}$$

$$\overline{x}_{jk} \in \{0,1\}, j,k = 1,2,...,n \tag{17}$$

where $C_{1k}(x,y) \triangleq \overline{C}_k(\overline{x})$ for $\mu = i = 1$, and the only optimization variable $\overline{x} = [x_{j1k}]_{j,k=\overline{1,n}} \triangleq [\overline{x}_{jk}]_{j,k=\overline{1,n}}$ is the sub-matrix of $x$. Please note that the task release dates $r_j$ do not depend on $\overline{x}$ for $\mu = 1$. Consequently, minimization of (11) on $\overline{x}$

with constraints (12)-(17) is the well-known task scheduling problem $1|r_j|\sum C_j$, which is strongly NP-hard (Lenstra et al., 1977). ■

Such a high computational complexity of the bi-criteria optimization problem justifies the search for heuristic algorithms that allow obtaining solutions in an acceptable time. Therefore, an evolutionary approach has been adopted. The general scheme of the NSGA II (Deb et al., 2002) with some improvements have been used and presented in the next section.

## 3. Solution algorithm

As the problem includes two contrary criteria, a multi-objective optimization approach is applied.

### *3.1 Encoding*

Chromosome $E = (e_1^{exe}, ..., e_\mu^{exe}, e_{\mu+1}^{s}, ..., e_{\mu+n}^{s}, e_{\mu+n+1}^{task}, ..., e_{\mu+2n}^{task}) \triangleq ((e^{exe}), (e^{s}), (e^{task}))$ encoded as a three-part sequence represents a candidate solution ($y$, $x$). It contains binary values $e^{exe}$ and $e^{s}$ along with integer values from the set $J$ for $e^{task}$. The binary values $e_i^{exe}$ directly represent $y_i$ and allow calculating $m$. The value of $\sum_{l=1}^{t} e_l^{exe} = i$ calculated for every $e_t^{exe} = 1$ points out the index $i$ of executor deployed at the location $t$. The following mappings decode the optimization variable $y$ and the number of launched locations (executors) $m$:

$$y = f_y(E) = \left[ e_i^{exe} \right]_{i=\overline{1,\mu}}^{\mathrm{T}}, \tag{18}$$

$$m = f_m(E) = \sum_{t=1}^{\mu} e_t^{exe}. \tag{19}$$

The $m-1$ '1's in $e^{s}$ located at positions $w_i$, $i = 1, 2,..., m-1$ and referred to as indices of separation specify the division of the set of all tasks into $m$ subsequences of tasks assigned to individual executors. Namely, the *i*th index of separation indicates the first position of the (*i*+1)th subsequence of $e^{task}$ representing tasks assigned to the executor deployed at the opened location pointed out by the (*i*+1)th in order '1' in $e^{s}$. The first subsequence of tasks in $e^{task}$, assigned to the executor pointed out by the first '1' in order in $e^{exe}$, starts at $e_{\mu+n+1}^{task}$. Consequently, the three-dimensional binary matrix $x$ can be retrieved from $E$ by the mapping:

$$x = f_x(E) = \left[ x_{j,i+1,k} = \begin{cases} 1, & j = e_{\mu+n+w_i+k-1}^{task},\ k = 1,2,...,w_{i+1} - w_i, \\ & w_0 = 1,\ i = 0,1,...,m-1, \\ 0, & \text{otherwise} \end{cases} \right] \tag{20}$$

The adopted method searches for solutions iteratively and independently for each subproblem that takes into consideration the fixed number of executors. It starts with the generation of an initial population $\tilde{S}_i = \left\{ \tilde{E}_{il} \right\}_{l=\overline{1,\alpha_i}}$ of size $\alpha_i$, $i = 1,2,...,\mu$, which cardinality $\left| \tilde{S}_i \right| = \alpha_i$ is equal for each subproblem.

The solutions are created by uniformly setting the values within the genome. The higher than $\vartheta$ Hamming distance between each pair of chromosomes is assumed. This diversification strategy helps to avoid premature convergence during the initial iterations.

---

**Algorithm** ALG_BC

---

**Require:** Data of BC_ScheLoc: $J$, $B$, $A$, $\mu$, $p_j$, $\rho_j$, $v_j$, $c_i$, and parameters of the algorithm: $\varphi_m$, $\varphi_c$, $\alpha_i$, $\bar{\alpha}$, $\tilde{\alpha}$, $\Gamma$, $\Gamma_{max}$.

**Ensure:** Set of non-dominated solutions $S_{\mathrm{PF}}$ and the Pareto front PF.

1: $i := 1$, $z := 1$, $\tilde{z} := 1$, $S_M(iz) := \varnothing$

2: **while** $i \le \mu$ **do**

3: Generate the initial population $\widetilde{S_i}$ of $\alpha_i$ solutions for exactly $i$ executors and set $\overline{S_i}(0) := \varnothing$

4: **while** $iz < \Gamma_{max}$ or $\tilde{z} < \Gamma$ **do**

5: Perform a non-dominated sorting and crowding-distance assignment.
6: Use a crowded comparison operator for a selection.
7: Generate the set $\overline{S}_i(z)$ of offsprings using the crossover (CPC-2/OX2) and mutation (SIM/SM) operators.
8: **for each** $l$ th element in $\overline{S}_i(z)$ that is non-dominated by any element in $S_M(iz)$ **do**
9: $S_M(iz) := S_M(iz) \cup \overline{E}_{il}(z)$
10: **end for**
11: **if** $\left|\overline{S}_i(z)\right| = \left|\overline{S}_i(z-1)\right|$ **then** $\tilde{z} := \tilde{z} + 1$ **else** $\tilde{z} := 1$ **end if**
13: **if** $\Gamma_{\max} \bmod z = 0$ or $\tilde{z} := \Gamma$ **then** set $i := i + 1$, $z := 1$ and **go to 2, end if**
16: **end while**
17: Set $z$: = $z$ : = $z + 1$.
18: **end while**
19: **return** $S_{PF} = \left\{\left(x_{\mathrm{PF,l}} = f_x\left(E_{L,l}\right), y_{\mathrm{PF,l}} = f_y\left(E_{Lu}\right)\right), l = 1,2,...,L(iz)\right\}$

and $PF = \left(\left(q^{(1)}(x_{\mathrm{PF,l}}, y_{\mathrm{PF,l}}), q^{(2)}(y_{\mathrm{PF,l}})\right) \triangleq (q_l^{(1)}, q_l^{(2)}) : q_{l-1}^{(1)} < q_l^{(1)}, l = 1,2,...,L(iz)\right)$

### *3.2 Selection, Crossover, and Mutation*

The presented method preserves the NSGA II structure. A selection process is based on the crowded comparison operator described in Deb (2001). The chromosomes undergo a parallel crossover with the use of two different operators with the probability $\varphi_c = 0.95$ (Grefenstette, 1986). For the binary parts, the Count-Preserving Crossover (CPC-2) operator has been used (Hou and Chang, 2004). This approach allows preserving $\sum_{i=1}^{\mu} y_i$ bits equal to 1 and hence the chosen number of executors and the right number $\sum_{i=1}^{\mu} y_i - 1$ of the indices of separation. For the integer part, the Order-based Crossover Operator (OX2) recombines solutions (Syswerda, 1991). The procedure is repeated until it achieves $\tilde{\alpha}$ solutions, which are passed on for further processing.

After the crossover, the mutation is applied to each offspring individually with a small probability $\varphi_m = 0.01$ (Hesser and Männer, 1990). Due to the importance of the order of tasks performed on a single executor, the Simple Inversion Mutation operator (SIM) has been applied for the integer part of $E$ (Holland 1975). The SIM operator selects randomly two cut points in the sequence, and it reverses the subsequences between them. Additionally, the binary parts in $E$ are randomly modified by the Swap Mutation (SM) operator (Oliver et al., 1987). The randomly selected single gene is swapped with another one that contains the different value to preserve the constant number of bits equal to one.

The crossover and mutation operators produce $\overline{\alpha}$ new offsprings that are stored in $\overline{S}_i(z) = \left\{\overline{E}_{il}(z)\right\}_{l=\overline{1,\overline{\alpha}}}$ where $z$ denotes an index of iteration. The used chromosome's form does not need to perform any repair strategies after the modifications made within $E$. The achieved results are aggregated in the long-term memory $S_{\mathrm{M}}(iz) = \left\{E_{\mathrm{M},l}(iz)\right\}_{l=\overline{1,L(iz)}}$ (elitist strategy), where $L(iz)$ is the size of Pareto front in the *iz*th iteration if and only if they are nondominated in the Pareto's sense. For all random-based operations, the Well Equidistributed Long-period Linear pseudorandom number generator named WELL44497a was applied (Panneton et al., 2006).

### *3.3 Stop condition*

Finally, the stop condition is fulfilled if there is no improvement in solutions representing the Pareto front through $\Gamma$ consecutive iterations. Moreover, the parameter $\Gamma_{max}$ has been introduced to restrict the maximal number of iterations for each subproblem. Consequently, the NSGA II-based heuristic solution algorithm referred to as Alg_BC is proposed for solving BC_ScheLoc.

### *3.4 Tunning*

The tuning is a two-step procedure, and the values of different parameters are fixed before the execution of the heuristics. In the first step, the algorithm's parameters are divided into two separate groups: chosen for tuning and non-tuned. During the second step, the grid search method is applied for the selection of the best set of parameters. Four parameters of Alg_BC with restricted domains of their possible values have been selected for tuning: $\alpha_i, \overline{\alpha} \in \{100, 150, ..., \mathbf{350}, ..., 500\}$, $\tilde{\alpha} \in \{0.6\alpha_i,$

$0.7\alpha_i$, $\mathbf{0.8}\alpha_i$, $0.9\alpha_i\}$, $\Gamma \in \{20, 30, ..., \mathbf{50}, ..., 100\}$, where bold types mark the default values. Analogously, the following default values of non-tuned parameters have been assumed: $\Gamma_{max} = 2500$, $\varphi_c = 0.95$, $\varphi_m = 0.01$, $\vartheta = \lfloor 0.1(\mu + n) \rfloor$. The tunings have been carried out for the following values of the problem's data: $n \in \{50,100\}$, $\mu \in \{4, 6, 8, 10, 15, ..., 35\}$, $\sigma = 0.25$, $\rho_j = 0$, $\gamma = 3.5$. The Cartesian coordinates $a_j^{(1)}$, $a_j^{(2)}$, $b_i^{(1)}$, $b_i^{(2)} \in [0,1000]$, the processing times $p_j \in \{10, 11, ..., 100\}$, and the costs of locations $c_i \in \{70, 71, ..., 139\}$ have been randomly generated from these domains according to the uniform distribution. We have assumed the Euclidean distance both in the tuning procedure and in further computational experiments. In consequence, the following values have been obtained: $\alpha_i = 350$, $\bar{\alpha} = 300$, $\tilde{\alpha} = 0.8\alpha_i$, $\Gamma = 40$.

## 4. Computational experiments

The evaluation of Alg_BC is the most significant purpose of the conducted computational experiments. First of all, it consists in the comparison of Pareto fronts generated by Alg_BC with similar outcomes created by the Matlab solver for small instances. The results are provided in Subsection 4.2, which follows the foundations for all experiments specifying the used dataset and quality indices adequate for evaluation and comparison. The sensitivity analysis completes the experiments. It allows checking the influence of the critical parameters on the results delivered by Alg_BC. All the computations have been made using a PC with the Intel Core Xeon W CPU 14-core processor of 2.5GHz equipped with 128GB of RAM. The algorithm Alg_BC has been implemented in the R language.

### *4.1 Foundations of computational experiments*

We used in the experiments a dataset of instances for different locations of tasks $a_j$, available locations for executors $b_i$, task processing times $p_j$, and costs of executor locations $c_i$, which values were randomly generated according to the uniform distribution from the sets $a_j^{(1)}, a_j^{(2)}, b_i^{(1)}, b_i^{(2)} \in [0,1000]$, $p_j \in \{20, 21, ..., 50\}$, and $c_i \in \{70, 71, ..., 139\}$. The speeds of task movement $v_j$ were analogously generated according to the formula $v_j = 100\gamma / p_j$ for $\gamma = 0.7, 1.4, 3.5, ..., 7.0$ where **3.5** stands for the default value. According to changes in the value of parameter $\gamma$, such a form of generation allows examining different ratios between $p_j$ and $r_j$. The used range enabled us to change the release dates $r_j$ from $p_j$ when $\gamma = 0.7$ to $0.1p_j$ when $\gamma = 7$. It is assumed that $a_j$ and $b_i$ are located inside the square with side length 1000, and 700 is the mean distance between them. We also considered the domains $\{1, 2, ..., 100\}$ and $\{1, 2, ..., 35\}$ for $n$ and $\mu$, respectively, with $n = 100$ and $\mu = 30$ as the default values. Moreover, $\rho_j = 0$ for all the tasks. In all computations, the default values were assumed if other ones were not specified in descriptions.

Apart from a Pareto front $\mathrm{PF} \triangleq ((q_l = (q_l^{(1)}, q_l^{(2)}) : q_{l\text{-}1}^{(1)} < q_l^{(2)})_{l=\overline{1,L}}$ generated by Alg_BC, which is an $L$-element sequence of points, we distinguish during the performed computational experiments three specific Pareto fronts: super Pareto front $\mathrm{PF_S}$, true Pareto front $\mathrm{PF_T}$, and Pareto anti front $\mathrm{PF_A}$.

The point $q_\lambda = \arg\min_{q_l \in \mathrm{PF}} \sqrt{(q_l^{(1)})^2 + (q_l^{(2)})^2}$ is distinguished within $\mathrm{PF}$. It is the closest point to $(0,0)$ according to the Euclidean distance. The values of its entries express the trade-off between both criteria. The $L_\mathrm{S}$-element super Pareto front $\mathrm{PF_S} \triangleq ((\mathrm{q}_{\mathrm{S},l}=(\mathrm{q}_{\mathrm{S},l}^{(1)}, \mathrm{q}_{\mathrm{S},l}^{(2)}) : \mathrm{q}_{\mathrm{S},l\text{-}1}^{(1)} < \mathrm{q}_{\mathrm{S},l}^{(2)})_{l=\overline{1,L_\mathrm{S}}}$ is determined for a given problem instance by the elimination of all dominated points from the union of PFs generated during 25 runs of Alg_BC. The point $q_{\mathrm{S},\lambda}$ is calculated analogously, like $q_\lambda$ for PF. The true Pareto front $\mathrm{PF_T} \triangleq ((\mathrm{q}_{\mathrm{T},l}=(\mathrm{q}_{\mathrm{T},l}^{(1)}, \mathrm{q}_{\mathrm{T},l}^{(2)}) : \mathrm{q}_{\mathrm{T},l\text{-}1}^{(1)} < \mathrm{q}_{\mathrm{T},l}^{(2)})_{l=\overline{1,L_\mathrm{T}}}$ is the result of an exact algorithm performed by the Matlab solver and applied for the assessment of Alg_BC. $\mathrm{PF_S}$ is used to have a $\mathrm{PF_T}$. Namely, the solver yield solutions for two single-criterion optimization problems for each $q_{\mathrm{S},l}$, $l = \overline{1, L_\mathrm{S}}$, i.e., $\min_x q^{(1)}(x; y)$ subject to $q^{(2)}(y) \le q_{\mathrm{S},l}^{(2)}$ and $\min_y q^{(2)}(y)$ subject to $q^{(1)}(x; y) \le q_{\mathrm{S},l}^{(1)}$ where $q_{\mathrm{S},l}^{(1)}$ and $q_{\mathrm{S},l}^{(2)}$ are the Cartesian coordinates of the point $q_{\mathrm{S},l}$ in $\mathrm{PF_S}$. The $L_\mathrm{T}$-point front $\mathrm{PF_T}$ is the result, and the point $q_{\mathrm{T},\lambda}$ applies accordingly. Such defined $\mathrm{PF_S}$ can also be used in experiments for bigger instances when $\mathrm{PF_T}$ is not known. Both mentioned Pareto fronts are graphically presented in Fig.1. The Pareto anti front

comprises all points belonging to all obtained results from 25 independent runs of Alg_BC, which do not dominate any other found point.

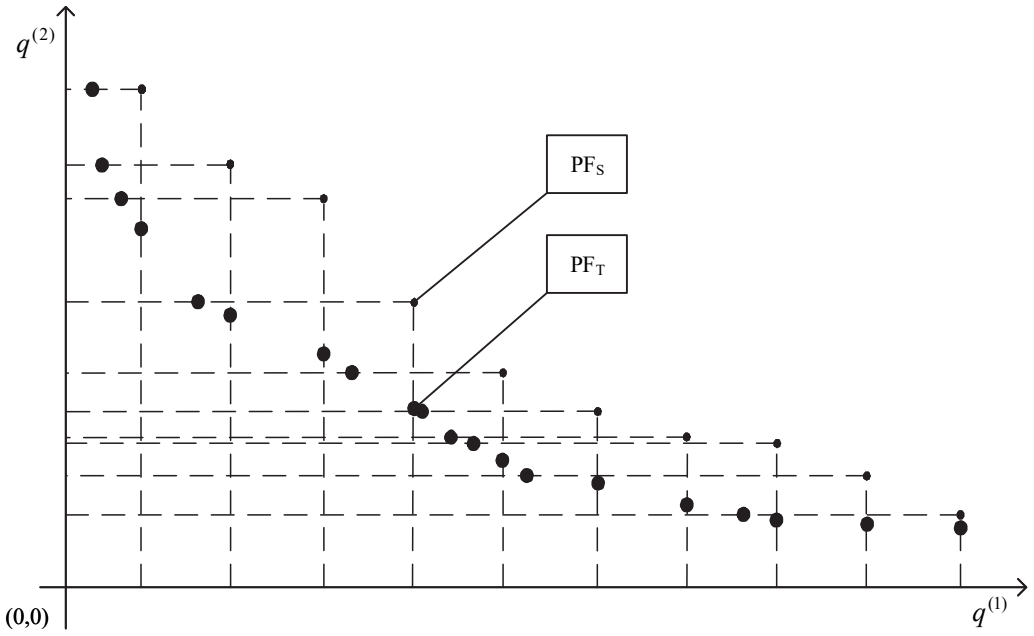


**Fig. 1.** Pareto fronts $\mathrm{PF}_S$ and $\mathrm{PF}_T$.

Besides the Euclidian distance

$$Dist = \min_{q_l \in \mathrm{PF}} \sqrt{(q_l^{(1)})^2 + (q_l^{(2)})^2}, \tag{21}$$

between $(0,0)$ and $q_\lambda$, we proposed a hypervolume-based quality index $I_{\mathrm{AH}}$ (Fig. 2), which is the adjustment of a well-known hypervolume indicator $I_{\mathrm{H}}$ (Zitzler et al., 2007; Laszczyk and Myszkowski, 2019). Both indices use normalized values of criteria and measure the part of the criteria area weakly dominated by an evaluated Pareto front. Unlike $I_{\mathrm{H}}$ which is calculated for the fixed reference point (1, 1), the proposed index $I_{\mathrm{AH}}$ takes into consideration the smaller and more essential dominated area defined by the adequate $\mathrm{PF}_{\mathrm{T}}$. So, $I_{\mathrm{AH}}$ is figured out in relation to $\mathrm{PF}_{\mathrm{T}}$, and it enables us to know how much the evaluated PF differs from $\mathrm{PF}_{\mathrm{T}}$. It is necessary to stress that the maximum value of $q^{(1)}$ cannot be calculated accurately due to the strong NP-hardness of the task scheduling subproblem. In consequence, its upper bound estimation is applied, which results in undesirable enlargement of the dominated area. Such an effect is mitigated, not eliminated by the adequate definition of $I_{\mathrm{AH}}$ where some parts of the dominated area are cut off. The also calculated ratio $I_{\mathrm{AH}} / I_{\mathrm{T,AH}}$ makes the relation between areas dominated by PF and $\mathrm{PF}_T$ more meaningful and informs us how much of the area dominated by $\mathrm{PF}_{\mathrm{T}}$ is covered by the corresponding area for PF where $I_{\mathrm{T,AH}}$ is the index $I_{\mathrm{AH}}$ for $\mathrm{PF}_{\mathrm{T}}$. Values close to 1 are preferable. The analytic formula for $I_{\mathrm{AH}}$ is as follows

$$I_{\mathrm{AH}} = \sum_{l \in S_{\mathrm{AH}}} (\max_{k \in 1, L_{\mathrm{T}}} \overline{q}_{\mathrm{T},k}^{(1)} - \overline{q}_l^{(1)})(\overline{q}_{l-1}^{(2)} - \overline{q}_l^{(2)}) \tag{22}$$

where $\overline{q}_0^{(2)} = \max_{k=1,L_{\mathrm{T}}} \overline{q}_{\mathrm{T},k}^{(2)}$, $\overline{q}_l^{(i)} = q_l^{(i)} / q_{\max}^{(i)}$, $\overline{q}_{\mathrm{T},k}^{(i)} = q_{\mathrm{T},k}^{(i)} / q_{\mathrm{T},\max}^{(i)}$, $i = 1,2$ are normalized values of both criteria for PF, and $S_{\mathrm{AH}} = (l \in \{1, 2, ..., L\}, \overline{q}_l^{(i)} \le \max_{\eta=1,L_{\mathrm{T}}} \overline{q}_{\mathrm{T},\eta}^{(i)}, i = 1,2)$ is a $L_{\mathrm{AH}}$-element sequence of indices sequentially indicating elements of PF belonging to the area determined by $\mathrm{PF}_{\mathrm{T}}$. The values normalizing both criteria are calculated as $q_{\max}^{(1)} = \sum_{i=1}^{\mu} c_i$, and $q_{\max}^{(2)}$ is the greatest value of $q^{(2)}$ found during all Alg_BC and Matlab solver runs for current problem instance. The computational time *Time* additionally evaluates Alg_BC.

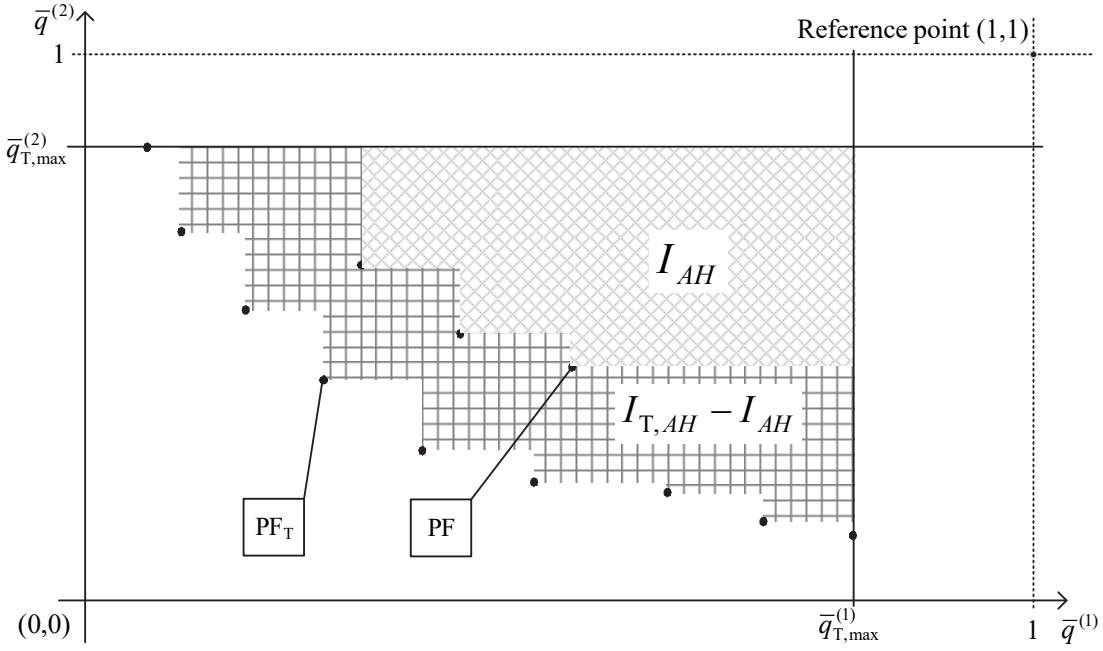


**Fig. 2.** Graphics for $I_{\mathrm{AH}}$ and $I_{\mathrm{T,AH}}$

*4.2 Results of the experiments*

The evaluation of Alg_BC was based mainly on the comparison of selected quality indices calculated for PFs and generated by Alg_BC with the corresponding indices calculated for $\mathrm{PF_T}$ by the solver. The values of both criteria and quality indices are provided in Tables 2-5, where Avg, Min, and Max mean respectively average, minimum, and maximum values obtained through 25 independent runs of Alg_BC for the same instance. The results of Alg_BC equivalent with the corresponding Matlab outcomes are marked in the tables by the bold types.

For the majority of analyzed instances, Alg_BC was able to find during its 25 runs the same point $q_\lambda$ as the solver, Table 2. The average values of $q_\lambda^{(1)}$, $q_\lambda^{(2)}$ are worse than $q_{\mathrm{T},\lambda}^{(1)}$, $q_{\mathrm{T},\lambda}^{(2)}$ no more than 17%, and 54%, respectively. The worst outcomes do not exceed 26% and 62% for $q_{\mathrm{T},\lambda}^{(1)}$ and $q_{\mathrm{T},\lambda}^{(2)}$, respectively. The worst results for the instances (15, 8), (25, 4) show the weaker performance of Alg_BC for a comparatively big number of possible locations $\mu$ or when $\mu$ is small compared with the number of tasks $n$. Undoubtedly, the small values of $\mu$ support the determination of cost criterion values $q_\lambda^{(2)} = q_{\mathrm{T},\lambda}^{(2)}$. For such instances, executors have been deployed at all available locations, and only such a decision has allowed reaching the best values of $q_\lambda^{(1)}$.

**Table 2**

Dependence of $q_\lambda$ and $q_{\mathrm{T},\lambda}$ on $n$ and $\mu$ (bold type points the results of Alg_BC equivalent to those obtained by the solver)

| $(n,\mu)$ | $q_{\mathrm{T},\lambda}^{(1)}$ | $q_\lambda^{(1)}$ | | | $q_{\mathrm{T},\lambda}^{(2)}$ | $q_\lambda^{(2)}$ | | |
|---|---|---|---|---|---|---|---|---|
| | | Avg | Min | Max | | Avg | Min | Max |
| (15, 4) | 1433 | 1565 | **1433** | 1636 | 398 | **398** | **398** | **398** |
| (15, 6) | 1253 | 1322 | **1253** | 1413 | 515 | 630 | **515** | 644 |
| (15, 8) | 1131 | 1192 | **1131** | 1330 | 496 | 759 | **496** | 800 |
| (20, 4) | 2265 | 2487 | **2265** | 2640 | 398 | **398** | **398** | **398** |
| (20, 6) | 1922 | 2028 | **1922** | 2157 | 515 | 639 | **515** | 644 |
| (20, 8) | 1635 | 1804 | 1715 | 1887 | 671 | 791 | **671** | 800 |
| (25, 4) | 3163 | 3680 | 3363 | 3959 | 398 | **398** | **398** | **398** |
| (25, 6) | 2744 | 2986 | 2785 | 3123 | 527 | 639 | **527** | 644 |
| (25, 8) | 2339 | 2564 | 2408 | 2670 | 800 | **800** | **800** | **800** |

**Table 3**
Dependence of $Dist$ and $Dist_T$ on $n$ and $\mu$

| $(n, \mu)$ | $Dist_T$ | $Dist$ | | | $Dist / Dist_T$ | | |
|---|---|---|---|---|---|---|---|
| | | Avg | Min | Max | Avg | Min | Max |
| (15, 4) | 1487 | **1487** | **1487** | **1487** | **1.000** | **1.000** | **1.000** |
| (15, 6) | 1409 | 1411 | **1409** | 1417 | 1.001 | **1.000** | 1.005 |
| (15, 8) | 1321 | 1340 | **1321** | 1385 | 1.014 | **1.000** | 1.048 |
| (20, 4) | 2300 | **2300** | **2300** | **2300** | **1.000** | **1.000** | **1.000** |
| (20, 6) | 2027 | 2127 | **2027** | 2251 | 1.049 | **1.000** | 1.111 |
| (20, 8) | 1821 | 1970 | 1893 | 2049 | 1.082 | 1.040 | 1.125 |
| (25, 4) | 3114 | 3144 | **3114** | 3225 | 1.009 | **1.000** | 1.036 |
| (25, 6) | 2819 | 2929 | 2858 | 3060 | 1.039 | 1.014 | 1.085 |
| (25, 8) | 2472 | 2686 | 2538 | 2788 | 1.087 | 1.027 | 1.128 |

The values of the akin single-valued index $Dist$ are presented in Table 3. The conclusions are similar. Again, Alg_BC was able to find during its 25 runs the results of a similar quality as the solver. The difference is worse by 9% on average but no more than 13%. However, the performance of Alg_BC deteriorates as $n$ and $\mu$ grow. The hypervolume-based index $I_{AH}$, which values are presented in Table 4, allows us to assess Alg_BC for the Pareto front PF as a whole, not only its selected points. The difference between the average value $I_{AH}$ and $I_{T,AH}$ does not exceed 7%. The analysis of ratio $I_{AH} / I_{T,AH}$ shows that the dominated area of PF covers, on average, at least 93% of the corresponding area determined for $PF_T$. The worst result was obtained for (25, 8) where the mentioned worst coverage is about 91%. It is necessary to stress that multiple runs of Alg_BC can substantially improve its quality. The values of $Time_T$ and $Time$ provided in Table 5 are incomparable as the former ones reach dozens of hours for launched small instances, unlike the latter ones, which do not exceed several seconds. Due to the exponential time complexity, the usage of the solver is impossible for higher instances, even if the generated results are non-operational but tactical or strategic decisions where the computational time is not the most essential.

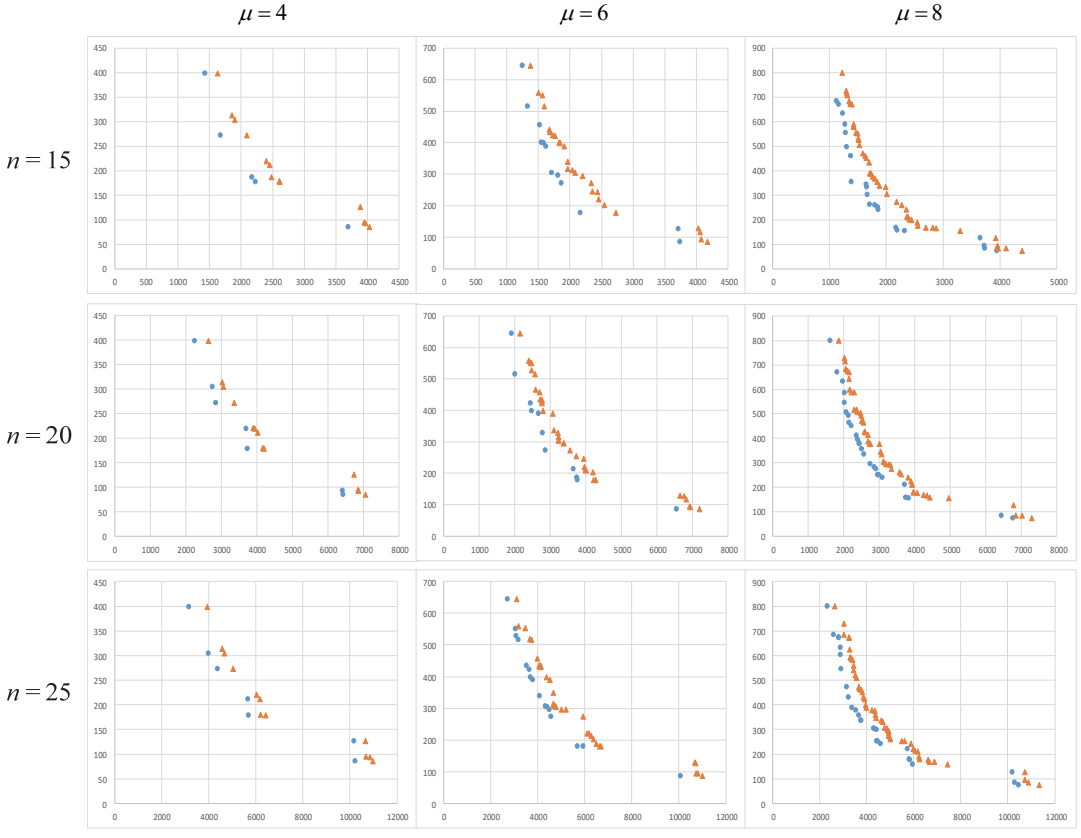


**Fig. 3.** True fronts $PF_T$ (dots) and anti fronts $PF_A$ (triangles) for different $n$ and $\mu$

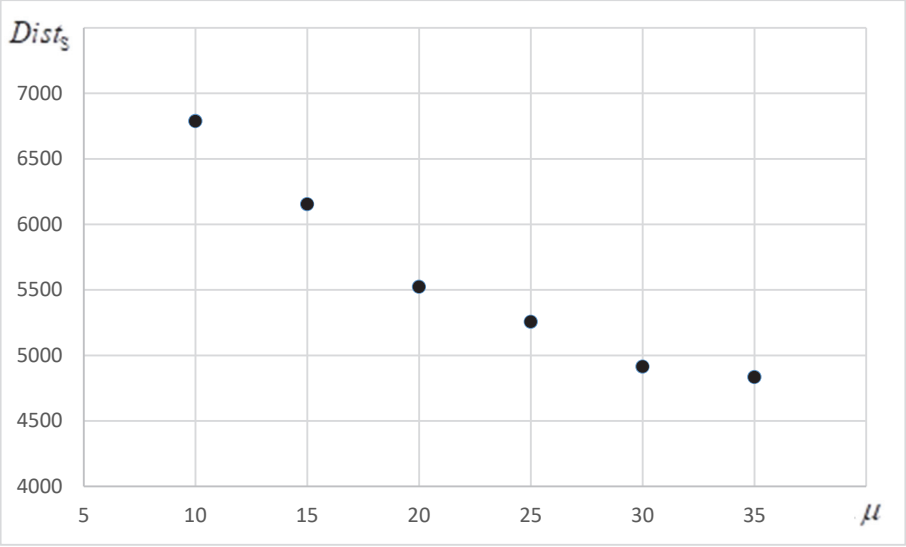


**Fig. 4.** Dependence of $Dist_S$ on $\mu$

The evaluation of Alg_BC quality is presented in Fig. 3 in the form of graphics of $PF_T$ (in dots) and $PF_A$ (in triangles) for the same instances $(n, \mu)$ as in Tables 2-5. The area between both fronts is the zone where we can expect to have PFs generated by Alg_BC. The size of this zone is rather small, which indicates the decent performance of Alg_BC. This size, however, is more significant for the higher number of tasks $n$. Moreover, it is worth noting that the size of the fronts grows when increasing the number of available executors $\mu$.

**Table 4**
Dependence of $I_{AH}$ and $I_{T,AH}$ on $n$ and $\mu$

| $(n, \mu)$ | $I_{T,AH}$ | $I_{AH}$ | | | $I_{AH} / I_{T,AH}$ | | |
|---|---|---|---|---|---|---|---|
| | | Avg | Min | Max | Avg | Min | Max |
| (15, 4) | **0.503** | **0.503** | **0.503** | **0.503** | **1.000** | **1.000** | **1.000** |
| (15, 6) | **0.523** | 0.520 | 0.514 | **0.523** | 0.994 | 0.982 | **1.000** |
| (15, 8) | 0.542 | 0.533 | 0.527 | **0.542** | 0.983 | 0.972 | **1.000** |
| (20, 4) | 0.599 | 0.585 | 0.582 | **0.599** | 0.977 | 0.972 | **1.000** |
| (20, 6) | 0.624 | 0.603 | 0.594 | **0.624** | 0.966 | 0.952 | **1.000** |
| (20, 8) | 0.688 | 0.651 | 0.642 | 0.660 | 0.946 | 0.933 | 0.959 |
| (25, 4) | 0.741 | 0.721 | 0.716 | 0.726 | 0.973 | 0.966 | 0.979 |
| (25, 6) | 0.790 | 0.755 | 0.741 | 0.762 | 0.956 | 0.938 | 0.965 |
| (25, 8) | 0.818 | 0.763 | 0.751 | 0.770 | 0.933 | 0.918 | 0.941 |

**Table 5**
Dependence of $Time$ and $Time_T$ on $n$ and $\mu$

| $(n, \mu)$ | $Time_T$ [h] | $Time$ [ms] | | |
|---|---|---|---|---|
| | | Avg | Min | Max |
| (15, 4) | 11.1 | 3044 | 3013 | 3101 |
| (15, 6) | 16.9 | 3793 | 3670 | 3886 |
| (15, 8) | 24.4 | 4990 | 4849 | 5033 |
| (20, 4) | 12.0 | 3921 | 3858 | 4101 |
| (20, 6) | 23.5 | 5288 | 5103 | 5312 |
| (20, 8) | 26.0 | 7010 | 6803 | 7180 |
| (25, 4) | 10.4 | 4783 | 4601 | 4902 |
| (25, 6) | 27.9 | 6002 | 5887 | 6121 |
| (25, 8) | 31.1 | 8192 | 8033 | 8301 |

We have also examined the performance of Alg_BC for bigger instances from the used dataset for various values of parameters $\mu$ and $\gamma$. The results are provided in Tables 6, 7, and Figs. 4 and 5. They show that in the range up to 20 $q_{S,\lambda}$ and $Dist_S$ are more sensitive to changes of $\mu$. The further increase of the possible number of locations for executors only slightly alters the outcomes of Alg_BC. The computational times, which values $Time_{S,avg}$ are averaged for 25 independent runs, grow nearly

linearly when increasing $\mu$. The comparison of these times with adequate results from Table 5 shows that the growth of $n$ and $\mu$ results in the acceptable increase of $Time_{S,avg}$. For example, the value of $Time_{S,avg}$ for (100, 35) is about 1.5 minutes. The value of the parameter $\gamma$ allows us to differentiate the relationship between the processing times $p_j$ and the release dates $r_j$. Smaller $\gamma$ closer $r_j$ to $p_j$. The dependence of $q_{S,\lambda}$ on $\gamma$ is not evident; however, the changes in $q_{S,\lambda}^{(1)}$ are more substantial in comparison to $q_{S,\lambda}^{(2)}$ which fluctuates slightly. The distances $Dist_S$ of $q_{S,\lambda}$ to (0, 0) are similar for $\gamma \in [3.5,7]$ so for $0.1p_j \le r_j \le 0.5p_j$. The higher values of $r_j$, closer to $p_j$, can move $q_{S,\lambda}$ away from (0, 0). The essential differences between $Time_{S,avg}$ for different $\gamma$ have not been noticed. The results of other conducted sensitivity analyses are not provided. Among others, it concerns the dependencies of $q_{S,\lambda}$, $Dist_S$, and $Time_{S,avg}$ on the number of tasks $n$ and the dispersion among location costs $c_i$ when no apparent relationships have been observed.

**Table 6**
Dependences of $q_{S,\lambda}$, $Dist_S$, and $Time_{S,avg}$ on $\mu$ for $n$=100

| $\mu$ | $q_{S,\lambda}^{(1)}$ | $q_{S,\lambda}^{(2)}$ | $Dist_S$ | $Time_{S,avg}$ [ms] |
|---|---|---|---|---|
| 10 | 6706 | 985 | 6788 | 29411 |
| 15 | 6003 | 1357 | 6154 | 38851 |
| 20 | 5198 | 1868 | 5523 | 52031 |
| 25 | 4671 | 2410 | 5256 | 65312 |
| 30 | 4364 | 2260 | 4914 | 80244 |
| 35 | 4206 | 2383 | 4834 | 94377 |

**Table 7**
Dependences of $q_{S,\lambda}$, $Dist_S$, and $Time_{S,avg}$ on $\gamma$ for $n$=100

| $\gamma$ | $q_{S,\lambda}^{(1)}$ | $q_{S,\lambda}^{(2)}$ | $Dist_S$ | $Time_{S,avg}$ [ms] |
|---|---|---|---|---|
| 0.7 | 13041 | 1933 | 13183 | 80445 |
| 1.4 | 7796 | 2014 | 8051 | 80933 |
| 2.1 | 6387 | 2080 | 6717 | 79953 |
| 2.8 | 5160 | 2245 | 5849 | 80774 |
| 3.5 | 4364 | 2260 | 4914 | 80331 |
| 4.2 | 4202 | 1943 | 4629 | 79901 |
| 4.9 | 4021 | 2171 | 4569 | 80001 |
| 5.6 | 3812 | 2183 | 4392 | 80023 |
| 6.3 | 3693 | 1851 | 4421 | 80221 |
| 7 | 3611 | 2154 | 4204 | 80588 |

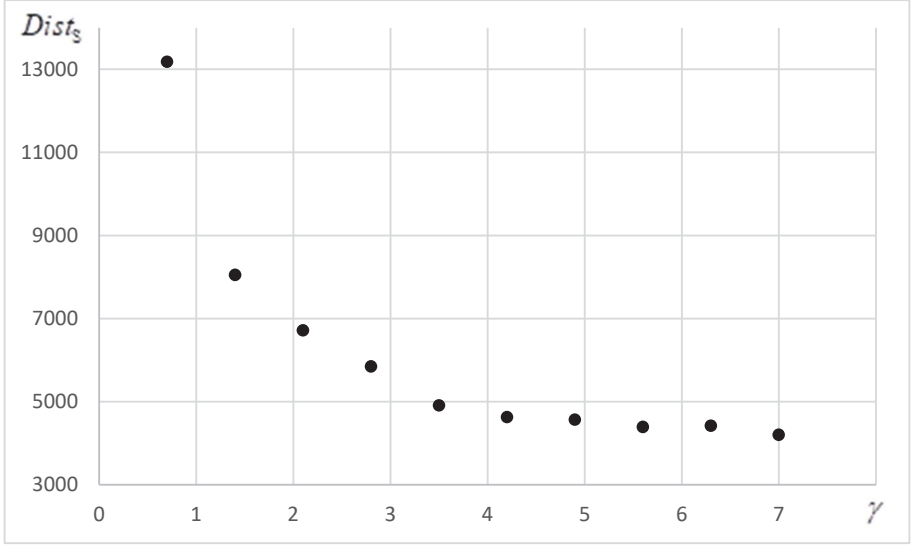


**Fig. 5.** Dependence of $Dist_S$ on $\gamma$

## 5. Case study

The combined location and task scheduling have some real-world applications mentioned in Section 1. The evacuation of people from danger zones is one of them (Heßler, 2017). However, it is the subject of broader interest and has been investigated in many other works, e.g., in Sbayti and Mahmassani (2006), see also Osman and Ram (2017), Amideo and Scaparra (2017), Abounacer et al. (2014) for other cases and methods. Referring to these and similar works, we demonstrate in this section the advantage of modelling a specific planning problem of evacuation in the framework of the considered problem and the usefulness of applying Alg_BC for the determination of evacuation plans.

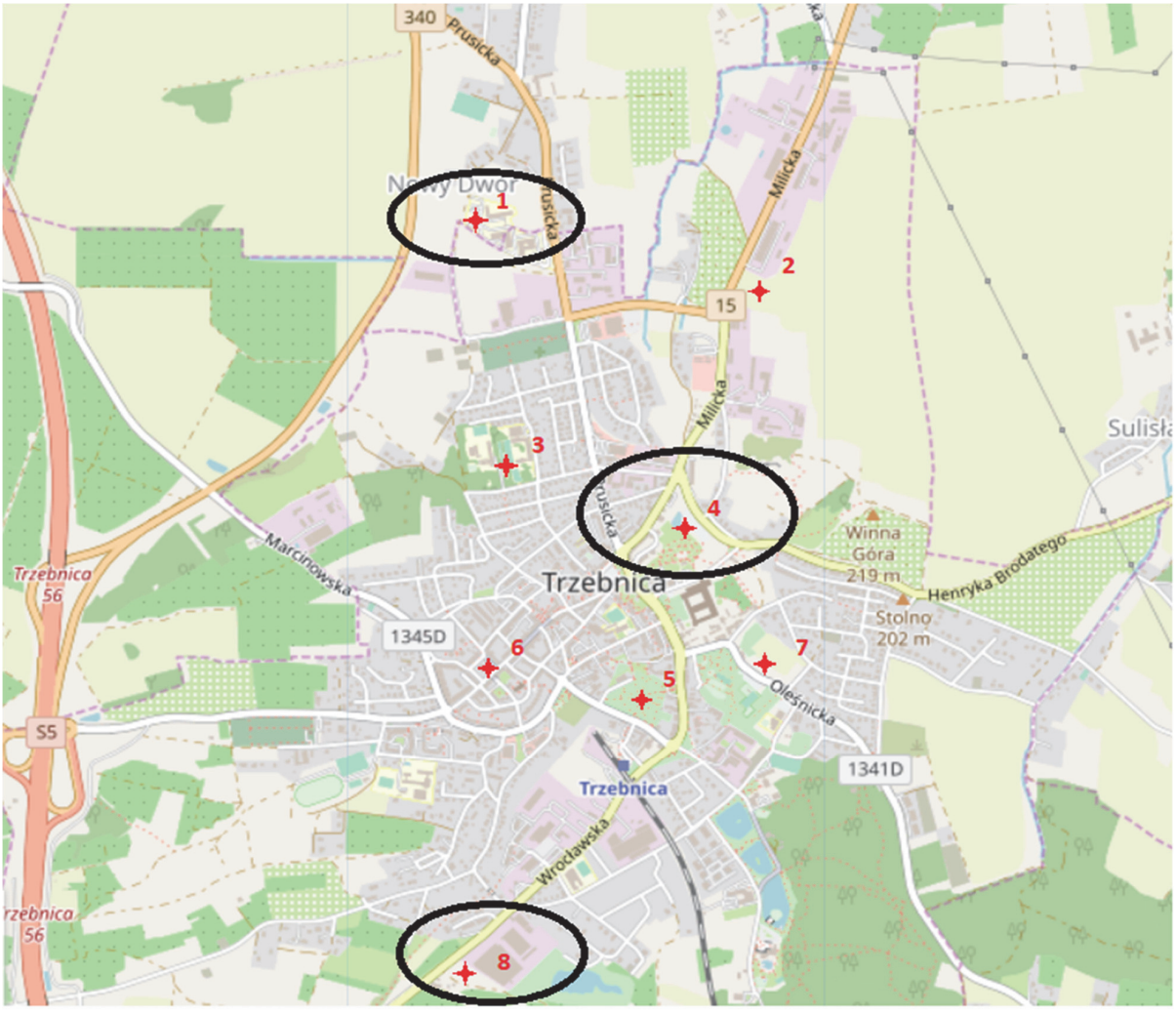


**Fig. 6.** Available locations of landing zones (red crosses) with marked in black chosen locations.

The case study deals with the planning of evacuation of citizens living in the Polish medium-sized city Trzebnica located in the south-west part of Poland. Some local requirements for evacuation planning specify the usage of helicopters as transportation facilities. Therefore, it is necessary to propose locations for adequate landing zones together with the assignment of home estates to the zones along with the order of evacuation. The possible locations of landing zones are shown in Fig. 6. The purpose of planning is not only to provide the solution that ensures the prompt leaving of the danger area but also to limit investment costs. Both conditions are fulfilled, and all decisions can be made when modeling the problem as joint location and scheduling. The city has been divided into thirty areas ($n = 30$). Moreover, eight prospective sites for the landing of helicopters have been distinguished ($\mu = 8$). When establishing the data, we assumed that the execution times $p_j$ are proportional to the size of the population from respective areas, and each person needs 1 min to be served. Then the ready times have been calculated as $\rho_j = 3 \ln p_j$. The values of other problem parameters adequate to the case study are presented in Table 8.

**Table 8**
Assumed data for the evacuation problem

| Parameter | Values |
|---|---|
| $n$ | 30 |
| $\mu$ | 8 |
| $p_j$ [min] | 100, 200, 200, 300, 300, 600, 400, 800, 300, 150, 600, 200, 200, 300, 300, 100, 400, 600, 100, 100, 200, 500, 150, 300, 200, 50, 400, 300, 500, 300. |
| $(a_{xj}, a_{yj})$ [km] | (2.347, 1.501), (1.585, 1.756), (1.541, 1.477), (1.349, 1.328), (1.226, 1.611), (1.072, 1.644), (0.970, 1.843), (0.961, 1.376), (1.142, 1.198), (0.914, 1.060), (0.751, 1.146), (0.628, 1.299), (0.504, 1.329), (0.539, 1.513), (0.399, 1.324), (0.429, 0.994), (0.439, 1.661), (0.555, 2.171), (0.658, 1.999), (0.892, 2.029), (0.872, 2.557), (1.019, 2.625), (0.956, 3.010), (0.666, 2.781), (1.471, 3.685), (1.219, 2.184), (1.139, 2.539), (1.254, 1.922), (1.099, 1.388), (1.234, 1.261). |
| $\rho_j$ [min] | 13.8, 15.9, 15.9, 17.1, 17.1, 19.2, 18.0, 20.1, 17.1, 15.0, 19.2, 15.9, 15.9, 17.1, 17.1, 13.8, 18.0, 19.2, 13.8, 13.8, 15.9, 18.6, 15.0, 17.1, 15.9, 11.7, 18.0, 17.1, 18.6, 17.1. |
| $v_j$ [km/h] | 3.75, 3.53, 3.53, 3.33, 3.33, 2.86, 3.16, 2.61, 3.33, 3.64, 2.86, 3.53, 3.53, 3.33, 3.33, 3.75, 3.16, 2.86, 3.75, 3.75, 3.53, 3.00, 3.64, 3.33, 3.53, 3.87, 3.16, 3.33, 3.00, 3.33. |
| $(b_{xi}, b_{yi})$ [km] | (1.919, 1.333), (1.768, 2.238), (1.368, 1.469), (1.245, 2.052), (0.854, 1.894), (0.956, 1.378), (0.975, 2.333), (0.290, 1.279). |
| $c_i$ [mln \$] | 0.15, 0.2, 0.5, 0.4, 0.8, 0.9, 0.3, 0.35 |

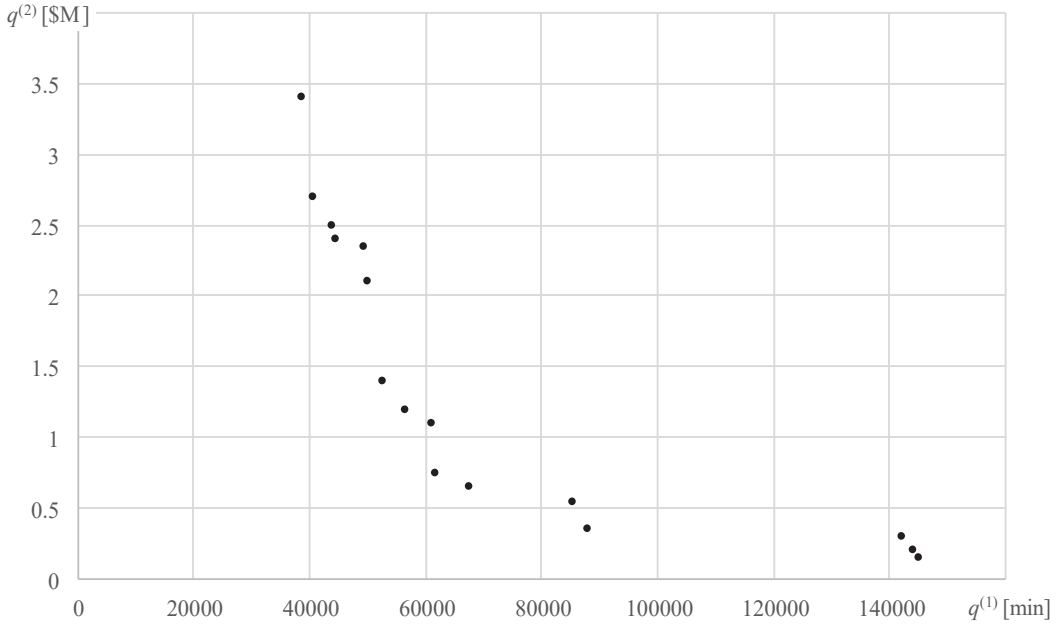


**Fig. 7.** The obtained Pareto front PF

The solution obtained by Alg_BC is presented in Tables 9, 10, and Fig. 7. Table 10 and Fig. 7 demonstrate the coordinates of sixteen PF points representing sixteen equivalent solutions, which details are provided in Table 9. For the readability, the sequences of tasks for individual executors (launched locations) substitute matrix $x$. The last row of Table 9 informs us how many times the individual locations $b_i$ occur in PF.

The variety of Alg_BC's outcomes facilitates a planner/a user to choose an appropriate solution.

In the case of an emergency, a planner is forced to minimize the evacuation time $q^{(1)}$ at the cost of $q^{(2)}$. In consequence, he/she would be ready to accept 7 out of 8 locations for landing zones ($m = 7$).

However, a planner is usually unable to accept that many landing zones due to a limited budget. The analysis of PF provides information on which landing zones are suggested the most in the solutions. Namely, Zone #1 ($b_1$) is most often proposed. Zone #4 ($b_4$) appears in the results in the second place.

Zones #2, #7, #8 ($b_2$, $b_7$, $b_8$) have the next similar numbers of occurrences in PF. However, taking into consideration the geographical distribution of landing zones, a planner should choose Zone #8 ($b_8$) as the third landing zone. It is doubtful that more landing zones would be built for such a small city. It is worth noting that the landing zones could be launched by turns

in a more extended period starting from Zone #1. The versions with one, two, or three zones are illustrated in Table 11, which like Table 9 presents the sequences of tasks.

**Table 9**

Sequences of tasks assigned by Alg_BC to $m$ launched locations $b_i$ for all points $l = \overline{1,16}$ from PF, and the number #o of individual locations occurred in PF (last row)

| $l$ | $m$ | locations |
|---|---|---|
| 1 | 7 | $b_1$ :(9, 2, 1, 14, 25, 24, 7, 23); $b_3$ :(18); $b_4$ :(15, 28, 3); $b_5$ :(12, 22, 20, 16); $b_6$ :(11, 29, 17, 10, 30); $b_7$ :(4, 26, 21, 13, 5, 8); $b_8$ :(27, 6, 19) |
| 2 | 7 | $b_1$ :(15); $b_2$ :(22, 27, 9, 1, 6); $b_3$ :(17, 25); $b_4$ :(28, 24, 2, 30, 7, 23, 21); $b_5$ :(4, 18, 8); $b_7$ :(3, 19, 11, 12, 20, 13); $b_8$ :(29, 5, 16, 10, 14, 26) |
| 3 | 6 | $b_1$ :(11, 12, 4, 3, 1, 30); $b_2$ :(9, 6); $b_3$ :(28, 7, 27, 20, 5, 8, 23, 26, 10); $b_4$ :(21, 29, 19, 2, 25); $b_6$ :(24, 22, 18, 13, 15); $b_8$ :(14, 16, 17) |
| 4 | 6 | $b_1$ :(4, 22, 1); $b_2$ :(15, 27, 25, 28); $b_3$ :(30, 7, 21, 8, 11); $b_4$ :(26, 20, 6, 2, 17, 14, 12, 10); $b_5$ :(9, 23, 5, 13); $b_8$ :(3, 18, 24, 16, 19, 29) |
| 5 | 6 | $b_1$ :(27, 7, 4, 18, 11, 21, 12); $b_2$ :(1, 23, 10, 3); $b_3$ :(13); $b_4$ :(28, 29, 20, 24, 19); $b_5$ :(6, 26, 30, 5, 25, 15); $b_7$ :(8, 9, 17, 2, 22, 16, 14) |
| 6 | 5 | $b_1$ :(8, 3, 12, 6, 24, 29); $b_4$ :(23, 17, 7, 26, 1, 19, 27, 5, 21); $b_6$ :(18, 2, 20, 14, 13, 4, 22, 25); $b_7$ (11, 28, 16, 30, 15); $b_8$ :(9, 10) |
| 7 | 5 | $b_1$ :(22, 17, 4, 23, 15, 24, 26, 2, 28, 30); $b_2$ :(7, 18, 19, 29); $b_4$ :(20, 14, 12, 16, 6); $b_7$ :(21, 27, 5, 10, 25, 11); $b_8$ :(13, 3, 8, 1, 9) |
| 8 | 4 | $b_1$ :(2, 17, 9, 6); $b_4$ :(4, 25, 18, 26, 16, 20, 19, 15, 21, 22, 13, 23, 28); $b_7$ :(8, 7, 3, 14); $b_8$ :(11, 30, 29, 12, 10, 27, 5, 24, 1) |
| 9 | 4 | $b_1$ :(28, 7, 14, 26, 25, 13, 22); $b_2$ :(30, 23, 8, 1); $b_4$ :(21, 15, 9, 4, 27, 16, 5, 2, 11, 6, 17, 10); $b_8$ :(12, 3, 19, 20, 18, 29, 24) |
| 10 | 3 | $b_1$ :(30, 28, 2, 1, 22, 7, 20, 26, 21, 6); $b_2$ :(8, 12, 13, 10, 3, 11, 14, 15, 19, 4, 5, 16); $b_4$ :(27, 29, 25, 24, 23, 17, 9, 18) |
| 11 | 3 | $b_1$ :(2, 8, 18, 7, 17, 1, 12, 28, 29, 16, 6); $b_2$ :(15, 4, 20, 3, 5, 21, 23, 27); $b_7$ :(30, 22, 10, 19, 9, 25, 26, 11, 14, 24, 13) |
| 12 | 2 | $b_1$ :(28, 14, 10, 24, 13, 17, 11, 22, 8, 12, 19, 21, 5, 3); $b_4$ :(25, 23, 2, 15, 26, 30, 16, 1, 9, 27, 29, 4, 7, 20, 18, 625, 23, 2, 15, 26, 30, 16, 1, 9, 27, 29, 4, 7, 20, 18, 6) |
| 13 | 2 | $b_1$ :(30, 4, 14, 21, 27, 17, 15, 10, 26, 1, 13, 19, 6, 29, 8, 9, 22); $b_2$ :(23, 28, 7, 3, 12, 25, 11, 16, 18, 20, 5, 2, 24) |
| 14 | 1 | $b_7$ :(9, 15, 10, 26, 13, 12, 17, 2, 27, 4, 28, 11, 5, 3, 21, 19, 30, 1, 25, 7, 29, 24, 8, 22, 23, 6, 14, 16, 18, 20) |
| 15 | 1 | $b_2$ :(24, 14, 4, 28, 30, 5, 13, 26, 9, 16, 1, 15, 6, 20, 7, 10, 23, 17, 19, 22, 25, 27, 2, 12, 29, 21, 8, 18, 11, 3) |
| 16 | 1 | $b_1$ :(29, 17, 1, 30, 11, 20, 10, 12, 5, 19, 21, 2, 18, 3, 7, 13, 27, 24, 23, 14, 26, 22, 9, 8, 25, 4, 16, 28, 6, 15) |

**Table 10**

Elements of Pareto front PF

| $l$ | $q_l^{(1)}$ | $q_l^{(2)}$ | $l$ | $q_l^{(1)}$ | $q_l^{(2)}$ |
|---|---|---|---|---|---|
| 1 | 38775 | 3.40 | 9 | 60885 | 1.10 |
| 2 | 40564 | 2.70 | 10 | 61707 | 0.75 |
| 3 | 43866 | 2.50 | 11 | 67507 | 0.65 |
| 4 | 44364 | 2.40 | 12 | 85268 | 0.55 |
| 5 | 49218 | 2.35 | 13 | 87779 | 0.35 |
| 6 | 49914 | 2.10 | 14 | 141863 | 0.30 |
| 7 | 52398 | 1.40 | 15 | 144083 | 0.20 |
| 8 | 56490 | 1.20 | 16 | 145059 | 0.15 |

**Table 11**

Sequences of tasks, i.e., groups of evacuated citizens for different number *m* of launched locations

| $m$ | Locations $b_i$ |
|---|---|
| 1 | $b_1$ :(18, 3, 21, 4, 22, 20, 30, 10, 26, 1, 19, 5, 27, 23, 7, 12, 25, 9, 13, 28, 11, 2, 24, 14, 17, 6, 8, 16, 29, 15) |
| 2 | $b_1$ :(11, 7, 10, 22, 4, 16, 5, 8, 29, 23, 30, 14); $b_4$ :(28, 17, 15, 1, 9, 26, 3, 13, 12, 24, 21, 18, 2, 25, 19, 27, 6, 20) |
| 3 | $b_1$ :(29, 1, 14, 10, 4, 2, 20, 19, 11, 13, 15, 24); $b_4$ :(6, 9, 22, 28, 18, 30, 8, 3, 21); $b_8$ :(5, 12, 25, 17, 7, 26, 16, 27, 23) |

## 6. Final remarks

We investigate a bi-criteria integrated approach for solving the location and scheduling (ScheLoc). The case is considered with a non-fixed number of identical parallel executors (machines) as well as non-preemptive independent tasks.

Two criteria evaluating both a schedule of tasks and location costs have been used. To our best knowledge, this is the first work elaborated for the bi-criteria ScheLoc problem with the sum of task completion times as the scheduling criterion. We provide a heuristic evolutionary algorithm due to the strong NP-hardness of the resulting combinatorial optimization problem. The conducted computational experiments showed the usefulness of the proposed algorithm. In many cases, the results generated by the algorithm are close and consistent with the exact outcomes provided by the Matlab solver. The average deterioration of the results does not exceed 19% when using the most comprehensive hypervolume-based quality index.

The experience gained during the conducted computational experiment will allow us to improve the algorithm. In particular, the better procedure for avoiding local optima would be advantageous, especially for bigger instances of the problem. Moreover, the comparison of current results with outcomes of algorithms SPEA2 and MOEA/D usable for multi-criteria optimization problems are planned as future works. The usefulness of the Tabu Search metaheuristics and a memetic approach will also be verified. Moreover, it seems interesting to develop a bi-criteria approach for other particular task scheduling cases, including unrelated (different, task-sensitive) executors as task performers. The theme of evacuation also needs a more in-depth investigation and extension on other similar real-world cases.

## References


Abounacer R., Rekik M., & Renaud J. (2014). An exact solution approach for multiobjective location–transportation problem for disaster response. *Computers & Operations Research*, *41*, 83–93

Deb K. (2001). *Multi-objective optimization using evolutionary algorithms* (Vol. 16). John Wiley & Sons

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182-197.

Drezner, Z., & Suzuki, A. (2004). The big triangle small triangle method for the solution of nonconvex facility location problems. *Operations Research*, *52*(1), 128-135

Elvikis D., Hamacher H., & Kalsch M. (2007). *Scheduling and Location (ScheLoc): Makespan Problem with Variable Release Dates*. Report in Wirtschaftsmathematik, Technische Universität Kaiserslautern, Fachbereich Mathematik

Elvikis, D., Hamacher, H. W., & Kalsch, M. T. (2009). Simultaneous scheduling and location (ScheLoc): the planar ScheLoc makespan problem. *Journal of Scheduling*, *12*(4), 361-374.

Amideo, A. E., & Scaparra, M. P. (2017, September). A Scenario Planning Approach for Shelter Location and Evacuation Routing. In *International Conference on Optimization and Decision Science* (pp. 567-576). Springer, Cham.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. IEEE Transactions on systems, man, and cybernetics, 16(1), 122-128.

Hennes H., Hamacher H. (2002). *Integrated Scheduling and Location Models: Single Machine Makespan Problems*. Report in Wirtschaftsmathematik, Univ., Fachbereich Mathematik

Hesser, J., & Männer, R. (1990, October). Towards an optimal mutation probability for genetic algorithms. In *International Conference on Parallel Problem Solving from Nature* (pp. 23-32). Springer, Berlin, Heidelberg.

Heßler, C., & Deghdak, K. (2017). Discrete parallel machine makespan ScheLoc problem. *Journal of Combinatorial Optimization*, *34*(4), 1159-1186.

Holland, J. H. (1992). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press.

Hou, Y. C., & Chang, Y. H. (2004). Short Paper. *Journal of Information Science and Engineering*, *20*, 1019-1034.

Kalsch, M. T. (2009). Scheduling-location (ScheLoc) models, theory and algorithms. Verlag Dr. Hut.

Kalsch, M. T., & Drezner, Z. (2010). Solving scheduling and location problems in the plane simultaneously. *Computers & operations research*, *37*(2), 256-264.

Jakob, K. R. A. R., & Pruzan, P. M. (1983). The simple plant location problem: survey and synthesis. *European journal of operational research*, *12*, 36-81.

Laszczyk, M., & Myszkowski, P. B. (2019). Survey of quality measures for multi-objective optimization: Construction of complementary set of multi-objective quality measures. *Swarm and Evolutionary Computation*, *48*, 109-133.

Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. In *Annals of discrete mathematics* (Vol. 1, pp. 343-362). Elsevier.

Liu, M., Liu, X., Zhang, E., Chu, F., & Chu, C. (2019). Scenario-based heuristic to two-stage stochastic program for the parallel machine ScheLoc problem. *International Journal of Production Research*, *57*(6), 1706-1723.

Ławrynowicz, M., & Józefczyk, J. (2019, August). A memetic algorithm for the discrete scheduling-location problem with unrelated machines. In *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)* (pp. 158-163). IEEE.

Oliver, I. M., Smith, D., & Holland, J. R. (1987). Study of permutation crossover operators on the traveling salesman problem. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987.

Osman, M. S., & Ram, B. (2017). Routing and scheduling on evacuation path networks using centralized hybrid approach. *Computers & Operations Research*, *88*, 332-339.

Panneton, F., L'ecuyer, P., & Matsumoto, M. (2006). Improved long-period generators based on linear recurrences modulo 2. *ACM Transactions on Mathematical Software (TOMS)*, *32*(1), 1-16.

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., & Tarantilis, C. D. (2017). Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research*, *263*(3), 737-754.

Pasandideh, S. H. R., Niaki, S. T. A., & Abdollahi, R. (2018). Modeling and solving a bi-objective joint replenishment-location problem under incremental discount: MOHSA and NSGA-II. *Operational Research*, 1-32.

Piasecki, B. (2018). *Application of AI-based algorithms for joint problem of task scheduling and deployment of executors (in Polish)*. Master's thesis, Wroclaw University of Science and Technology. Poland

Piasecki, B., & Józefczyk, J. (2018). Evolutionary algorithm for joint task scheduling and deployment of executors. *Automation of Discrete Processes. Theory and Applications, Silesian University of Technology, 1*, 169-178.

Pinedo, M. (2012). Scheduling: theory, algorithms and systems development (Vol. 29). Springer-Verlag NY

Prodhon, C., & Prins, C. (2014). A survey of recent research on location-routing problems. *European Journal of Operational Research*, *238*(1), 1-17.

Rajabzadeh, M., Ziaee, M., & Bozorgi-Amiri, A. (2016). Integrated approach in solving parallel machine scheduling and location (ScheLoc) problem. *International Journal of Industrial Engineering Computations, 7*(4), 573-584.

Rostami, M., & Bagherpour, M. (2020). A lagrangian relaxation algorithm for facility location of resource-constrained decentralized multi-project scheduling problems. *Operational Research*, 1-41.

Saldana, J., & Suznjevic, M. (2015). *Qoe and latency issues in networked games*.

Sbayti, H., & Mahmassani, H. S. (2006). Optimal scheduling of evacuation operations. *Transportation Research Record, 1964*(1), 238-246.

Scholz, D. (2011). Deterministic global optimization: geometric branch-and-bound methods and their applications (Vol. 63). Springer Science & Business Media.

Shahabi, M., Tafreshian, A., Unnikrishnan, A., & Boyles, S. D. (2018). Joint production–inventory–location problem with multi-variate normal demand. *Transportation Research Part B: Methodological*, *110*, 60-78.

Syswerda, G. (1991). Scheduling optimization using genetic algorithms. Handbook of genetic algorithms.

Wesolkowski, S., Francetić, N., & Grant, S. C. (2014, July). TraDE: Training device selection via multi-objective optimization. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2617-2624). IEEE.

Zitzler, E., Brockhoff, D., & Thiele, L. (2007, March). The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 862-876). Springer, Berlin, Heidelberg.