ISSN 1392–124X (print), ISSN 2335–884X (online) INFORMATION TECHNOLOGY AND CONTROL, 2016, T. 45, Nr. 3

# Hybrid Metaheuristic Method for Solving a Multi-Period Emergency Service Location Problem

**Stefan Mišković**

*Faculty of Mathematics, University of Belgrade,*
*Studentski trg 16/IV, 11 000 Belgrade, Serbia*
*e-mail: stefan@matf.bg.ac.rs*

**Zorica Stanimirović**

*Faculty of Mathematics, University of Belgrade,*
*Studentski trg 16/IV, 11 000 Belgrade, Serbia*
*e-mail: zoricast@matf.bg.ac.rs*

crossref http://dx.doi.org/10.5755/j01.itc.45.3.14041

**Abstract**. This study deals with the problem of establishing the network of emergency service units. The goal of the basic problem proposed in the literature is to locate certain number of units at given discrete points of the region and to allocate cities to established units, in order to balance the load of established emergency units. Having in mind that emergency units work in shifts, we extend the basic model to a multi-period model and involve additional constraints on the number of units to be located. Since, in practice, the number of emergency incidents varies on daily or monthly basis, we consider the uncertainty of the number of incidents and propose a robust integer programming formulation of the multi-period model, which controls the deviation of objective value under uncertainty of input data. In order to solve both deterministic and robust variant of the problem, we design an efficient hybrid metaheuristic method based on combination of Particle Swarm Optimization method (PSO) and Reduced Variable Neighborhood Search (RVNS). Computational results show that the proposed hybrid PSO-RVNS method quickly reaches all known optimal solutions obtained by CPLEX solver, and provides solutions for instances that remained out of reach of CPLEX. In the single-period case, PSO-RVNS outperforms existing metaheuristic method from the literature in the sense of CPU time. Short running times of PSO-RVNS and high-quality solutions indicate the efficiency of the proposed hybrid metaheuristic approach when solving the considered problem. Results presented in this study may help security experts and emergency managers to design an efficient and sustainable emergency system.

**Keywords**: emergency system; facility location problem; robust optimization; particle swarm optimization; reduced variable neighborhood search.

## 1. Introduction

In this paper, we consider a generalization of the problem of establishing the network of emergency service units that was introduced by Stanimirović et al. [33]. The authors of [33] deal with the problem of designing the network of the Police Special Forces Units (PSFUs) in the Republic of Serbia. A set of potential locations for establishing PSFU units is given, and a set of cities to be allocated to established units. Each city is assigned the number of criminal acts within its area, which was obtained from statistical data over past years. The number of the PSFUs to be located is limited by a given constant, and according to the most common situation in practice, it is assumed that each city will be assigned to its closest established PSFU unit. The model proposed in [33] also involves penalties in the case that a city is assigned to a PSFU unit that lies outside given range that ensures an efficient PSFU reaction. Differently from studies that optimize emergency service systems from customers' point of view, i.e., minimize the maximum distance between customer and service provider [8, 9, 11, 19], Stanimirović et al. [33] deal with designing the network of emergency units from providers' point of view [5, 13, 26]. In practice, emergency units are required to travel longer distances than others to reach an assigned city, or to react in cities with higher number of incidents, and therefore, the workload of emergency units will be

most likely unequal. On the other side, all units are generally equally paid, regardless of amount of their workload during a shift. Therefore, the objective of the model proposed in [33] is to minimize the maximal load of a PSFU unit, while preserving the efficiency of the emergency system. In the literature, there are examples of location models that involve workload balance of facilities in the objective [1, 16, 20, 21]. The balanced workload of facilities may be used in long-term planning and designing public services, such as health-care systems, determining optimal locations of various public services within a city area or a region, finding optimal locations of telecommunication hubs, etc.

Since in practice, the number of incidents in each city may vary from average values obtained from statistical data, it is necessary to take into account demand uncertainties in emergency network model. Capturing data uncertainty in deterministic models may be achieved in different approaches that were proposed in the literature in past years [7, 19, 27, 31, 36]. Starting from deterministic PSFU location model, Stanimirović et al. [33] also proposed a robust optimization model that captures the uncertainty of the number of incidents in each city. More precisely, in [33] it is assumed that the coefficients representing the number of criminal incidents in are subject to uncertainty, and they are modeled as independent, symmetric and bounded random variables with unknown distribution.

In [33], both deterministic and robust models are tested by CPLEX 12.5 commercial solver on the set of real-life test instances including up to 165 cities and 234 potential PSFU sites, and different protection levels. Obtained solutions are analyzed by security experts from practical point of view. Since the largest problem instance remained out of reach of CPLEX 12.5 solver, the Stanimirović et al. [33] also designed an evolutionary-based algorithm (EA) that was enhanced with a local search method (LS). The proposed hybrid method EA-LS was benchmarked on the same real-life data set. The results of computational experiments presented in [33] showed that the proposed hybrid method EA-LS quickly reached all known optimal solutions and provided solutions for the instances unsolved to optimality by CPLEX 12.5 solver. The analysis on the impact of different protection levels on the objective value increase was also presented.

In this study, we extend the model presented in [33] to a multi-period problem of locating emergency units, having in mind that emergency units usually work in two or three shifts during the day. In addition, we impose the upper limit on the number of the available emergency units through all time periods and the lower limits on the number of active units within each considered period. We further allow the number of emergency incidents to vary within each time period and propose a robust formulation of the deterministic multi-period model.

We also design a novel hybrid optimization method for solving both deterministic and robust variants of the considered problem. The proposed hybrid method is designed as a combination of Particle Swarm Optimization method (PSO) and Reduced Variable Neighborhood Search (RVNS). In each iteration of the proposed PSO-RVNS method, a subset of solutions obtained by PSO, are used as initial solutions for the RVNS, in order to obtain further improvements. Parameters of the PSO-RVNS are fine-tuned in order to achieve the best performance of the algorithm.

The proposed hybrid PSO-RVNS approach is benchmarked on single-period real-life problem instances used in [33] and generated multi-period instances that are derived from single-period ones. Results obtained by th PSO-RVNS approach are compared with the results of EALS from [33] for solving deterministic and robust variant of the single-period problem. Computational results clearly indicate the superiority of the proposed PSO-RVNS method over existing EA-LS heuristic in the sense of CPU times for problem instances of larger dimension. For multi-period case, most of the tested instances remained out of reach for CPLEX solver, while the proposed PSO-RVNS converges steadily to its best solutions, which coincide with optimal solutions obtained by CPLEX (in cases when optimal solutions are known). Short running times of the PSORVNS (even in the case of the largest considered instance) indicate the efficiency of the proposed hybrid approach.

The rest of paper is organized as follows. In Section 2, discrete optimization model of the multi-period problem is presented. In Section 3, the discrete model is extended to a robust optimization model that involves uncertainty of the number of emergency incidents. Hybrid PSORVNS metaheuristic method is described in Section 4. In Section 5, we present computational results obtained on real-life problem instances for single period case and compare them with the results of EA-LS method from [33]. Results of computational experiments for deterministic and robust multi-period case are also presented in Section 5. Finally, in Section 6 we give the summary of obtained results and some directions for future work.

## 2. Mathematical formulation of the multi-period emergency units location problem

Let $I$ represent the set of cities, and $J$ the set of potential locations for establishing emergency units. Let $T$ denote the set of considered time periods (work shifts of emergency units). As in [28], it is assumed that $I \subseteq J$ holds, meaning that an emergency unit may be located in a city itself. Each city $i \in I$ has assigned values $f_{ti} \geq 0$, $t \in T$ representing the average number of incidents in the city $i \in I$ in a time period $t \in T$.

For each pair of locations $i \in I$ and $j \in J$, the distance $d_{ij}$ is calculated as driving distance between two locations. It is assumed that between each pair of locations there is a direct link (road) connecting them. Let $c$ represents the maximal driving distance between a location of an emergency unit and a location of the

incident that allows the unit to react in a timely manner. For each $i \in I$ and $j \in J$, the following sets are defined:

- $C_{ij} = \{k \in J: d_{ik} \leq d_{ij}\}$, $i \in I$, $j \in J$ represents the set of potential locations $k \in J$ for which the distance from location $i$ is less than or equal to the distance from $i$ to $j$;
- $F_{ij} = \{k \in J: d_{ik} > d_{ij}\}$, $i \in I$, $j \in J$ stands for the set of potential locations $k \in J$ for which the distance from $i$ is greater than distance from $i$ to $j$;
- Note that $C_{ij} \cup F_{ij} = J$ and $C_{ij} \cap F_{ij} = \emptyset$ hold for all $i \in I$ and $j \in J$;
- $S_j = \{i \in I: d_{ij} \leq c\}$, $j \in J$ denotes the set of locations $i \in I$ for which the distance from location $j$ is less than or equal to $c$;
- $D_j = \{i \in I: d_{ij} > c\}$, $j \in J$ represents the set of locations $i \in I$ for which the distance from location $j$ is greater than $c$.
- Here $S_j \cup D_j = I$ and $S_j \cap D_j = \emptyset$ hold for all $j \in J$.

For each time period $t \in T$, we introduce an integer parameter $k_{t,min}$ representing the minimum number of emergency units to be located in time period $t \in T$. The total number of established units through all time periods is limited by integer $k_{max}$, where $k_{max} \geq \sum_{t \in T} k_{t,min}$.

As in [33], for each pair $i \in I$ and $j \in J$, a parameter $p_{ij}$ is involved, representing the penalty in the case that a location $i \in D_j$ is assigned to a unit at location $j$. In order to avoid excessively large penalty values, the value of parameter $p_{ij}$ is calculated as:

$$p_{ij} = \min\left\{\frac{|d_{ij}-c|}{c}, 1\right\}.$$

The proposed multi-period emergency service location model uses a continuous variable $z_{max}$ representing the objective function value, and two sets of binary variables that indicate location and allocation decisions. More precisely, binary variable $y_{tj}$, $t \in T$, $j \in J$ indicates if an emergency unit is established at location $j \in J$ in a time period $t \in T$:

$$y_{tj} = \begin{cases} 1, & \text{emergency unit is established at} \\ & \text{location } j \text{ in time period } t \\ 0, & \text{otherwise,} \end{cases}$$

while binary variable $x_{tij}$ indicates whether or not a police unit established at location $j \in J$ reacts on incidents in a city $i \in I$ for a time period $t \in T$:

$$x_{tij} = \begin{cases} 1, & \text{if in time period } t \text{ city i is assigned} \\ & \text{to unit at location } j, \\ 0, & \text{otherwise.} \end{cases}$$

The goal of the considered problem is to determine locations for emergency units, such that the maximal value of

$$\sum_{i \in S_j} f_{ti} x_{tij} + \sum_{i \in D-J} f_{ti}(1 + p_{ij}) x_{tij}$$

for all $t \in T$ and $j \in J$ is minimized.

Having in mind the problem's nature, it is assumed that emergency incidents in each city are handled by the nearest established emergency unit (single allocation scheme and closest-assignment rule). In the considered problem, no capacity restrictions and no fixed costs for establishing units are involved. It is also assumed that the performance rate of each emergency unit is the same.

Using the notation and conditions mentioned above, the mixed integer linear programming (MILP) formulation of the multi-period emergency units location problem may be written as:

$$\min z_{max} \quad (1)$$

such that

$$\sum_{j \in J} x_{tij} = 1 \quad \forall t \in T \quad \forall i \in I, \quad (2)$$

$$x_{tij} \leq y_{tj} \quad \forall t \in T \quad \forall i \in I \quad \forall j \in J, \quad (3)$$

$$y_{tj} \leq \sum_{k \in C_{ij}} x_{tik} \quad \forall t \in T \quad \forall i \in I \quad \forall j \in J, \quad (4)$$

$$\sum_{j \in J} y_{tj} \geq k_{min,t} \quad \forall t \in T, \quad (5)$$

$$\sum_{t \in T} \sum_{j \in J} y_{tj} \leq k_{max}, \quad (6)$$

$$\sum_{i \in S_j} f_{ti} x_{tij} + \sum_{i \in D_j} f_{ti}(1 + p_{ij}) x_{tij} \leq z_{max} \quad \forall t \in T \quad \forall j \in J, \quad (7)$$

$$x_{tij} \in \{0,1\} \; \forall t \in T \quad \forall i \in I \quad \forall j \in J, \quad (8)$$

$$y_{tj} \in \{0,1\} \quad \forall t \in T \quad \forall j \in J, \quad (9)$$

$$z_{max} \geq 0. \quad (10)$$

By objective (1) and constraint (7), the maximal load of established emergency units through all time periods is minimized. Each city is assigned to exactly one emergency unit location, which is ensured by (2). Constraints (3)–(4) denote that each city is assigned to its closest established unit. By constraints (5), lower bounds on the number of located units in each time period are given, while constraint (6) imposes the upper bound on the number of established locations through all time periods. Constraints (7) define lower bounds on the value of objective variable $z_{max}$. Variables $x_{tij}$ and $y_{tj}$ are binary (8)–(9), while variable $z_{max}$ is nonnegative (10).

In the single-period case ($|T| = 1$), index $t$ may be omitted from variables $x_{tij}$ and $y_{tj}$, $i \in I$, $j \in J$. Parameters $f_{ti}$ are then reduced to $f_i$, $i \in I$, while $k_{min,t}$ is reduced to $k_{min}$. If the lower bound for number of established units is set to zero ($k_{min} = 0$), the MILP model (1)–(10) reduces to the MILP model proposed by Stanimirović et al. in [33].

## 3. Robust variant of the problem

There are numerous examples of applying traditional optimization techniques when dealing with uncertainty in emergency service design, such as stochastic programming and optimization under probabilistic constraints [17–19, 27, 34, 36]. However, the difficulties in applying stochastic programming arise

when the exact distribution of input data is unknown, which often happens when modeling real-life emergency service problems. For optimization under probabilistic constraints, the problem usually arises when it is not possible to cover all scenarios that capture the distribution of input data, even if all scenarios are known. In this case, the size of resulting optimization model increases drastically as a function of the number of scenarios. This results in fact that optimal solution can not be found due to time or memory limits of computer resources.

Robust optimization [32] represents an alternative approach to stochastic programming and optimization under probabilistic constraints. It allows us to control the degree of conservatism of the solution, and it is computationally tractable both practically and theoretically, see [2, 3]. In robust models, random variables are modeled as uncertain parameters belonging to a convex or polyhedral uncertainty set, and the goal is to protect the system against the worst case within the uncertainty set [4].

In this study, we use robust optimization approach to optimize the emergency system in the worst-case situations that arise under uncertainty of number of emergency incidents. The goal of robust model is to protect the emergency system against the uncertainty of the number of incidents $f_{ti}$ that occur in city $i \in I$ in time period $t \in T$. Therefore, the number of incidents in a city $i \in I$ in time period $t \in T$ is modeled as an independent and bounded random variable, denoted as $\tilde{f}_{ti}$ and it is assumed that it has unknown distribution. Considering the nature of the problem, we are interested in cases when $\tilde{f}_{ti}$ may decrease or increase from the nominal values $f_{ti}$. However, without loss of generality, we may suppose that $\tilde{f}_{ti} \in [f_{ti}, f_{ti} + \hat{f}_{ti}]$, where $\hat{f}_{ti} \geq 0$, $i \in I$, $t \in T$. Note that it is enough to consider the asymmetric interval, since symmetric interval $[g_{ti} - \hat{g}_{ti}, g_{ti} + \hat{g}_{ti}]$ may be obtained from $[f_{ti}, f_{ti} + \hat{f}_{ti}]$ by introducting substitutions $g_{ti} = f_{ti} + \hat{g}_{ti}$ and $\hat{f}_{ti} = 2\hat{g}_{ti}$.

Let us observe the set $G = \{(t,i) \in T \times I : \hat{f}_{ti} > 0$ consisting of all pairs $(t,i) \in T \times I$ for which the number of incidents increases from the nominal value $f_{ti}$. For robustness purposes, we introduce protection level parameter $\Gamma \in [0, |G|] \cap \mathbb{N}$, which controls level of robustness in the objective. In the case of $\Gamma = 0$, we completely ignore the change in the number of incidents, while in the case of $\Gamma = |G|$, all possible changes in the number of incidents are considered. In general, for higher values of Γ, the level of robustness is increased at the expense of higher objective values [4].

Using the notation and assumptions from deterministic model (1)–(10), the robust mathematical model of the considered multi-period emergency units location problem is given as follows:

(1), under constraints (2)–(6), (8)–(10) and

$$q_{tj} \leq z_{max} \quad \forall t \in T \quad \forall i \in I, \tag{11}$$

where

$$q_{tj} = \sum_{i \in S_j} f_{ti} x_{tij} + \sum_{i \in D_j} f_{ti}(1 + p_{ij}) + \max_{P \subset G : |P| \leq \Gamma} \sum_{(t,i) \in P} \hat{f}_{ti} x_{tij} \tag{12}$$

We now formulate Theorem 1 that will be used to present mathematical formulation of robust multi-period emergency units location model.

**Theorem 1.** *Consider the nominal Integer Programming (IP) problem*

$$\min \sum_{j \in J} c_j x_j,$$
$$\sum_{j \in J} a_{ij} x_j \leq b_j \quad \forall i \in I,$$
$$x_j \in \{0,1\} \quad \forall j \in J.$$

*Let $J_i$, $i \in I$ represent the set of coefficients $a_{ij}, j \in J_i$ that are subject to uncertainty, i.e., $J_i = \{j \in J : \hat{a}_{ij} > 0\}$. Let integer parameters $\Gamma_i \in [0, |J_i|]$, $i \in I$ denote protection levels for the i-th constraint. The IP problem*

$$\min \sum_{j \in J} c_j x_j,$$
$$\sum_{j \in J} a_{ij} x_j + \max_{S_i \subseteq J_i : |S_i| \leq \Gamma_i} \sum_{j \in S_i} \hat{a}_{ij} x_j \quad \forall i \in I,$$
$$x_j \in \{0,1\} \quad \forall j \in J$$

*has an equivalent Mixed Integer Linear Programming (MILP) formulation:*

$$\min \sum_{j \in J} c_j x_j,$$
$$\sum_{j \in J} a_{ij} x_j + z_i \Gamma_i + \sum_{j \in J_i} p_{ij} \leq b_i \quad \forall i \in I,$$
$$z_i + p_{ij} \geq \hat{a}_{ij} y_j \quad \forall i \in I \quad \forall j \in J_i,$$
$$p_{ij} \geq 0 \quad \forall i \in I \quad \forall j \in J_i,$$
$$z_i \geq 0 \quad \forall i \in I,$$
$$x_j \in \{0,1\} \quad \forall j \in J.$$

It is easy to see that the presented theorem follows directly from the Theorem 1 given in [4]. Finally, having in mind Theorem 1, the robust variant of the multi-period emergency units location model may be formulated as:

(1), subject to (2)–(6), (8)–(10) and

$$\sum_{i \in S_j} f_{ti} x_{tij} + \sum_{i \in D_j} f_{ti}(1 + p_{ij}) x_{tij} + \Gamma z + \sum_{i \in G} r_i \leq z_{max} \quad \forall t \in T \quad \forall j \in J_i, \tag{13}$$
$$z + r_i \geq \hat{f}_{ti} x_{tij} \quad \forall t \in T \quad \forall i \in G \quad \forall j \in J, \tag{14}$$
$$z \geq 0, \tag{15}$$
$$r_i \geq 0 \quad \forall i \in G. \tag{16}$$

Note that the robust model (1), subject to (2)–(6), (8)–(10) and (13)–(16) represents a generalization of the robust model proposed in [33].

## 4. Proposed hybrid metaheuristic method

Hybrid metaheuristics showed to be promising approaches for solving numerous optimization prob-

lems [6, 35]. In the literature, one of the most popular strategies to develop a hybrid metaheuristic method is to combine a population-based approach and a variant of a local search heuristic [20, 22, 25, 28]. In general, population-based heuristics offer more facilities for exploration, while local search methods provide more capabilities for exploitation. However, it is important to achieve a good balance between exploitation and exploration strategies, such that the resulting hybrid method provides high-quality solutions in reasonably short running times [6, 25].

In this paper, we present a hybrid metaheuristic method that is obtained by combining a Particle Swarm Optimization (PSO) as a population-based heuristic, and a Reduced Variable Neighbourhood Search (RVNS) as a local search heuristic. The proposed PSO-RVNS works over population $N_r$, consisting of $|N_r|$ solutions. In each iteration of the algorithm, solutions are evaluated and ranked in respect to their objective values. The objective function calculation is designed and implemented such that PSO-RVNS successfully solves both deterministic and robust variant of the problem. The PSO method is applied only to less-quality solutions from $N_r$, while high-quality solutions $N_e \subset N_r$ are directly passed to the RVNS part of the hybrid method. The RVNS heuristic is applied to all solutions from the set $N_r$, looking for their improvements. Described steps are repeated until a stopping condition is satisfied. The described way of combining PSO and RVNS methods takes advantage of good sides of both constructive heuristics. The basic structure of the proposed PSO-RVNS hybrid method is presented in Algorithm 1.

**Algorithm 1.** The basic structure of the PSORVNS method

```
1: Read input
2: while stopping condition is satisfied do
3:     i ← i + 1
4:     Calculate objective function value of the solutions from N_r
5:     Choose the set of high quality solutions N_e from N_r
6:     Apply PSO to all solutions from the set N_r \ N_e
7:     for r ∈ N_r do
8:         Apply RVNS to solution r
9:     end for
10: end while
11: Write output
```

### 4.1. Representation of solutions

Each solution from the set $N_r$ is assigned an information on locations of established emergency units in each time period. It is represented by a binary array of length $N$, where $N = |T| \cdot |J|$. The binary array consists of $|T|$ segments of length $|J|$, where each segment $t \in \{1, \dots, |T|\}$ corresponds to one time period. If location $j$ is opened in a period $t$, the $((t-1) \cdot |J| + j)$-th bit in the array takes the value of 1, and 0 otherwise.

A solution is denoted as *feasible* if

- the number of bits in the $t$-th segment of its code is greater than or equal to $k_{t,min}$, and
- the total number of bits with the value of 1 is less than or equal to $k_{max}$.

For example, for $|T| = 3$, $|J| = 3$, $k_{1,min} = k_{2,min} = k_{3,min} = 1$ and $k_{max} = 4$, a solution to the problem is represented as $|100|010|010|$. In the first period, an emergency unit is established at location 1, while in the second and third period, emergency unit established at location 2. The solution is feasible, since the sum of all bits with the value of 1 is equal to 3 (which is less than $k_{max}$), while each time period contains at least $k_{t,min} = 1$ established unit.

### 4.2. Objective function calculation

The indices of locations with established emergency units for each time periods are read from the solution's code. After that, its objective function value in the case the deterministic model is calculated through the following steps:

**Step 1.** For each time period $t \in \{1, \dots, |T|\}$, create the subset of locations $E_t = \{j \in J :$ emergency unit is established at $j$ in time period $t\}$;

**Step 2.** For each city i and each time period $t$, assign the city $i$ is to a unit at location $j \in E_t$, such that the distance $d_{ij}$ is minimal. If there is more than one location $j \in E_t$ such that the distance $d_{ij}$ is minimal, randomly choose one of them;

**Step 3.** Let $w_{tj}$ represent the workload of an established unit at location $j \in E_t$. Set $w_{tj} = 0$;

**Step 4.** For all cities $i$ that are assigned to $j \in E_t$

a) if $d_{ij} \leq c$, increase $w_{tj}$ by the value of $f_{ti}$;

b) if $d_{ij} > c$, increase $w_{tj}$ by the value of $f_{ti}(1 + p_{ij})$;

**Step 5.** Set the objective function value to $\max\{w_{tj}: t \in T, j \in J\}$.

The complexity of objective function calculation in the deterministic case is equal to $O(|T| \cdot |I| \cdot |J|)$.

Once the objective function value of the solution to the deterministic case ($\Gamma = 0$) is calculated, the corresponding objective function value for the robust case ($\Gamma > 0$) may be obtained in an efficient manner by using the result of the following theorem.

**Theorem 2.** *Let $f_1, f_2, \dots, f_n, f_{n+1}$ be the nonnegative real parameters such that $f_1 \geq f_2 \geq \cdots \geq f_n \geq f_{n+1} = 0$ holds, and let $\Gamma$ be a positive integer parameter $\Gamma \in \{0, 1, \dots, n\}$. If $z + r_i \geq f_i$, $i = 1, \dots, n$ holds for $z, r_1, r_2, \dots, r_n \geq 0$, then the minimal value of function $F_\Gamma$: $\mathbb{R}^n \to \mathbb{R}$*

$$F_\Gamma(z, r_1, \dots, r_n) = \Gamma_z + \sum_{i=1}^{n} r_i$$

*is achieved for* $z = f_{\Gamma+1}$.

The proof of Theorem 2 can be found in Appendix A.

Let $F_\Gamma^{min}$ represent the objective function value of the robust model for the fixed protection level parameter $\Gamma > 0$. Let array $\hat{f}_{t_k i_k}$, $1 \le k \le M$, $M = |T| \cdot |I|$, represent a permutation of $\hat{f}_{ti}$, $t \in T$, $i \in I$ such that

$$\hat{f}_{t_1 i_1} \ge \hat{f}_{t_2 i_2} \ge \cdots \ge \hat{f}_{t_M i_M}. \quad (17)$$

From Theorem 2 and (13), it follows that

$$F_\Gamma^{min} = F_0^{min} + \sum_{k=1}^{\Gamma} \hat{f}_{t_k i_k}, \quad (18)$$

which allows us to easily calculate objective value of the solution to the robust model (with $\Gamma > 0$) when the objective value of the corresponding solution to the deterministic model ($\Gamma = 0$) is known. Note that similar conclusion may be applied for calculating objective function value for the robust variant of the problem proposed in [33].

### 4.3. Particle Swarm Optimization part

Particle swarm optimization method (PSO) is a population-based metaheuristic, based on the idea of swarm intelligence. Since 1995, when the concept of PSO was introduced by Kennedy and Eberhart [14], it has been applied to both continuous and discrete optimization problems in a wide range of areas. An overview of publications on applications of PSO to various optimization problems may be found in [29].

The PSO works over a swarm $X$ of particles moving in a $N$-dimensional search space ($N = |T| \cdot |J|$). Each swarm represents a candidate solution to the problem. PSO shares many similarities with evolutionary-based algorithms, but the main difference is that PSO includes no variation operators (e.g. crossover, mutation), and therefore, it is easier to implement compared to EA. In spite the fact that the structure of PSO is simpler than the structure of EA, the PSO still provides good and efficient diversification of solutions in the search space.

In this study, we design a variant of PSO that is used as population-based part of the proposed hybrid method. In each iteration of the hybrid algorithm, PSO is applied only to the set of non-elite solutions from the previous iteration. Therefore, the initial swarm of particles $X$ in each iteration is actually the set of non-elite solutions $N_r \backslash N_e$ from the previous iteration of the hybrid method. The percentage of elite individuals represents a parameter that is experimentally adjusted (see Section 5.1).

Each particle $i \in X$ is assigned vectors $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{v}_i \in \mathbb{R}^n$, $i \in X$. A $N$-dimensional binary vector $\mathbf{x}_i$ represents the current position of a particle $i \in X$ and corresponds to a candidate solution in the search space. The velocity of a particle is represented by a $N$-dimensional vector $\mathbf{v}_i$ with real coordinates that take values from interval $[v_{min}, v_{max}]$, where $v_{min}$ and $v_{max}$ are predetermined parameters. If a coordinate of the velocity vector $v_i$ exceeds $v_{min}$ or $v_{max}$, it is reset to $v_{min}$ or $v_{max}$, respectively. In addition, for each particle $i \in X$, its best visited position in $N$-dimensional binary vector $\mathbf{p}_i$ is memorized. The best and the second best position visited by whole swarm are saved in $N$-dimensional binary vectors $\mathbf{g}$ and $\mathbf{g}'$, respectively.

In the initialization phase, the positions of particles are set in accordance with non-elite solutions that are subject to PSO. More precisely, the position vector $\mathbf{x}_i$ of a particle $i \in X$ is equal to the binary code of a non-elite solution $i \in N_r \backslash N_e$. Coordinates of velocity vector $\mathbf{v}_i$, $i \in X$ are obtained by uniform distribution from $[v_{min}, v_{max}]$. Having in mind the nature of considered problem, in the proposed PSO implementation, $v_{min}$ and $v_{max}$ are set to 0 and 1, respectively. Initial values of vectors $\mathbf{p}_i$, $i \in X$, $\mathbf{g}$ and $\mathbf{g}'$ are also calculated in this step.

After the initialization phase, in each PSO iteration, a particle $i \in X$ successively adjusts its position $\mathbf{x}_i$ in respect to the best position $\mathbf{p}_i$ visited by itself, and the best position visited by the whole swarm $\mathbf{g}$. In addition, inspired by idea presented in [30], we use the second best global position $\mathbf{g}'$ when adjusting a particle's position. In [30] it is experimentally confirmed that the variant of PSO that uses the second best global position has better performance compared to the standard PSO. Therefore, in each iteration, coordinates of velocity change vector $\Delta v_{i,l}$ of a particle $i$ are calculated as:

$$\Delta \mathbf{v}_{i,l} = r_p c_p (\mathbf{p}_i - \mathbf{x}_i) + r_g c_g (\mathbf{g} - \mathbf{x}_i) + r_{g'} c_{g'} (g' - x_i), \; l = 1, 2, \dots, N, \; i \in X.$$

Note that velocity change vector of a particle also depends on a cognitive learning parameter $r_p$, and social learning parameters $r_g$ and $r_{g'}$.

Parameter $r_p$ represents the attraction that a particle will fly toward its own success, while parameters $r_g$ and $r_{g'}$ denote the tendency that a particle will be leaded by the success of the best and the second best positioned particle in whole swarm, respectively. In each PSO iteration, the values of parameters $r_p$, $r_g$ and $r_{g'}$ are chosen by uniform distribution from the interval (0;1). The values of $c_p$, $c_g$ and $c_{g'}$ are taken from paper by Shin and Kita [30]: $c_p = c_g = 1.5$, and $c_{g'} = 5$, since they are experimentally determined in [30]. Parameters $c_p$, $c_g$ and $c_{g'}$ have constant values through all PSO iterations.

In each PSO iteration, flying direction of a particle $i \in X$ is calculated as:

$$\mathbf{v}_{i,l} \leftarrow \begin{cases} 1, & \text{if } \mathbf{v}_{i,l} + \Delta \mathbf{v}_{i,l} > 1, \\ 0, & \text{if } \mathbf{v}_{i,l} + \Delta \mathbf{v}_{i,l} < 0, \\ \mathbf{v}_{i,l} + \Delta \mathbf{v}_{i,l} & \text{otherwise.} \end{cases}$$

After we obtain the flying direction of particle $i \in X$, a new particle position is calculated as $\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i$. Since we are dealing with discrete problem with binary variables, the velocity of a particle is associated with the probability that a bit in a particle's position vector will take the value of 1. Therefore, a sigmoid function $S(v) = 1/(1 + e^{-v})$ is used to normalize the coordinates of velocity vector into interval $[0,1]$, see [15]. A random number r is generated uniformly from $(0,1)$, and the coordinates of position vector $x_{i,l}, l = 1, 2, \dots, N$ of particle $i \in X$ are adjusted as follows:

$$\mathbf{x}_{i,l} \leftarrow \begin{cases} 1, & \text{if } r < (1 + e^{-v_{i,l}})^{-1}, \\ 0, & \text{otherwise.} \end{cases}$$

In this way, it is ensured that $x_{i,l} \in \{0, 1\}$, $l = 1, 2, \dots, N$ for the adjusted position of particle $i \in X$.

Note that it may happen the new particle position $\mathbf{x}_i$ may correspond to an infeasible solution. This situation will occur if the total number of bits $x_{i,l}$, $l = 1, 2, \dots, N$ with the value of 1 is greater than $k_{max}$, or if for some $t \in \{1, 2, \dots, |T|\}$, the number of bits $x_{i,l}$ with the value of 1 in the $t$-th segment of $\mathbf{x}_i$ is smaller than $k_{min,t}$. If the total number of bits with the value of 1 is $k$, where $k > k_{max}$, we randomly choose $k - k_{max}$ coordinates $x_{i,l}$ with the value of 1 and invert them to 0. Similarly, if number of bits with the value of 1 in the $t$-th segment is $k_t$, where $k_t < k_{min,t}$, exactly$k_{min,t} - k_t$ randomly chosen coordinates with the value of 0 are inverted to 1.

If a particle $i \in X$ has moved to a better position $\mathbf{x}_i$ compared to its best local position, vector $\mathbf{p}_i$ is updated with $\mathbf{x}_i$. If the new best local position is better than the best global one, the best global position $\mathbf{g}$ of the swarm is updated. The second best global position $\mathbf{g}'$ is also updated, if necessasry.

### 4.4. Reduced Variable Neighborhood Search part

Variable neighborhood search (VNS) is a metaheuristic method proposed by Mladenović and Hansen in [23]. The basic idea of VNS is systematic change of neighborhood within a local search. In general, VNS sequentially explores neighborhoods of the current solution looking for a better solution, and moves from the current solution to its neighbour if an improvement was made. The summary of VNS applications to various optimization problems and its hybrids with other optimization techniques can be found in [35].

Let $U_l(r)$ denotes the set of solutions belonging to the $l$-th neighborhood of the current solution $r$, where $l_{max}$ denotes the number of different neighborhood structures. Assuming that $l$ is initially set to 1, the basic VNS procedure typically consists of the following three phases:

- *Shake* – generate a random solution $r'$ from the $l$-th neighborhood of the current solution $U_l(r)$;
- *Local search* – apply a local search method starting from the randomly generated solution $r' \in U_l(r)$ and find a local optimum in $U_l(r)$;
- *Move* – if the local optimum is better than the current solution, move there and set $l \leftarrow 1$. Otherwise, set $l \leftarrow l + 1$.

These steps are repeated until $l = l_{max}$. The initial VNS solution is randomly generated, but it also may be obtained by the means of some other heuristic. In this way, it is ensured that VNS starts the search from a good quality solution.

In cases that local search is costly, the acceleration of VNS may be achieved by completely omitting the local search phase. This is the basic idea behind the variant of VNS, denoted as Reduced Variable Neighborhood search (RVNS). In RVNS, solutions are chosen at random in successive neighborhoods of the current solution, and the search is restarted each time when an improvement is obtained (see [12]). In this paper, we have used Reduced Variable Neighborhood Search instead of basic VNS, in order to achieve improvements within short running times. The results of preliminary experiments showed that the idea of omitting local search drastically reduced running time of the hybrid PSO-RVNS, while the solution quality was preserved.

The RVNS method is applied each iteration of the proposed hybrid algorithm. Let us consider a solution $r$ to the problem, and let $(t, j)$, $t \in T$, $j \in J$ denote a bit on $\big((t-1) \cdot |J| + j\big)$-th position in the code of solution r. Neighbourhood structures $U_l(r), l = 1,2,3$ used in this paper are as follows:

- $U_1(r)$ – the set of solutions obtained from solution r by swapping two different bits $(t, i)$ and $(t, j), i, j \in J, i \neq j$ in the same code's segment $t \in \{1, \dots, |T|\}$;
- $U_2(r)$ – the set of solutions obtained from solution r by swapping bits $(t_1, i)$ and $(t_2, j), i, j \in J$ belonging to two different code's segments $t_1, t_2 \in \{1, \dots, |T|\}$, $t_1 \neq t_2$;
- $U_3(r)$ – the set of solutions obtained from the solution r by inverting a randomly chosen bit $(t, j), i \in J$ from a code's segment $t \in \{1, \dots, |T|\}$.

The basic structure of the RVNS procedure used in this study is presented by Algorithm 2. In each iteration of the proposed hybrid method, the RVNS heuristic is applied 10 times to each solution $r$ from the population $N_r$. A neighbor solution $r'$ is considered better than solution $r$, if the objective function $f(r')$ is less than $c_r \cdot f(r)$. The value of parameter $c_r$ is adjusted by using analysis of variance (see Section 5.1).

When generating a neighbor solution $r$ in the Shaking phase of the RVNS, it is necessary to either invert the bit values from 0 to 1, or from 1 to 0 in the code of current solution $r$. Instead of recalculating objective function value of the generated neighbor from the beginning, we take into consideration only changes in objective value that resulted from inversion of the particular bit of the current solution $r$. The applied strategy significantly reduced computational time of the RVNS part.

**Algorithm 2.** RVNS method

```
for each solution r ∈ N_r do
    l ← 1
  while l ≤ l_max do
    Shaking: generate randomly solution r' ∈ U_l(r)
    if f(r') ≤ c_r · f(r) then
      r ← r'
      l ← 1
    else
      l ← l + 1
    end if
  end while
end for
```

Let us assume that the value of $\big((t-1)\cdot|J|+j\big)$-th bit in the code of current solution is changed from 1 to 0. It means that emergency unit at location $j$ in time period $t$ is removed. Therefore, the sum of $f_{ti}$ among all cities $i$ that were assigned to the unit at location $j$ in time period $t$ becomes 0. Each city $i$ that was assigned to the unit at location $j$ in time period $t$, is now being assigned to the established unit at its closest location $j'$ in the same time period. The value of the workload of the unit at location $j'$ is further updated. Since we only consider the set of cities that were assigned to the unit at location $j$ in time period $t$, the overall computational time of this step is reduced from $O(|T|\cdot|I|\cdot|J|)$ to $O\big(|J_j|\cdot|J|\big)$.

Similarly, let us assume that the value of $\big((t-1)\cdot|J|+j\big)$-th bit in the code of current solution is inverted from 0 to 1. This implies that emergency unit at location $j$ in time period $t$ is now established. For all cities $i\in I$, we check if the value $d_{ij}$ is less than $d_{ik}$, where $k$ is the location of the closest established emergency unit for city $i\in I$ in time period $t\in T$. If $d_{ij}<d_{ik}$ holds, the city $i\in I$ is now assigned to the unit at location $j$ in time period $t$. The workload of location $j$ is increased by $f_{ti}$, while the workload of location $k$ is decreased by the same value. As in previous case, the overall computational time of this step is $O(|I|)$.

### 4.5. Other aspects of the PSO-RVNS

Initial population, containing $|N_r|$ individuals, is generated by uniform distribution, thus providing a good diversity of the initial solutions. Infeasible individuals in the initial population are corrected to be feasible, as described in Section 4.3. The PSO method corrects infeasible solutions that might appear during PSO iterations (see Section 4.3), while RVNS improvement procedure is designed in such way that the feasibility of the solutions is preserved.

In the proposed PSO-RVNS, we use the elitism in generation replacement strategy. All individuals are ranked according to their objective function value, and the best fitted $|N_e|$ ones are selected as elite individuals. These individuals directly pass in the next PSO-RVNS generation, and therefore, they do not need re-calculation of the objective function value (since they have been evaluated in one of previous generations). In this way, we provide additional time-savings in total CPU time. Remaining (non-elite) individuals are replaced in the next generation. The percentage of elite individuals is denoted as $p_{el}$, which represents one of the parameters of the algorithm that is experimentally adjusted. The number of elite individuals $|N_e|$ is calculated as the product $p_{el}\cdot|N_r|$ rounded to the nearest integer.

If a duplicate individual appears during the algorithm's run, it is being removed by setting its objective value to $+\infty$. In this way, we tend to preserve the diversity of individuals and to prevent the premature convergence of the algorithm. The PSO-RVNS stops if the solution with the best objective value remains unchanged through $rep$ subsequent generations.

The number of individuals in the population $|N_r|$, the percentage of elite individuals in the population $p_{el}$, and the stopping criterion parameter $rep$ are experimentally determined by using analysis of variance (see Section 5).

## 5. Experimental analysis

All computational experiments in this study were carried out on an Intel i5-2430M 2.4 GHz with 8 GB RAM memory under Windows 7 operating system. The CPLEX 12.1 commercial solver was employed to obtain optimal solutions, if possible. The PSO-RVNS implementation was coded in C++ programming language. On each test instance, the proposed PSO-RVNS was run 15 times.

In order to benchmark PSO-RVNS method, we start from the real-life instances introduced in [33], which are obtained from the network of Police Special Forces Units (PSFUs) in the Republic of Serbia. These instances involve geographical positions locations of cities and potential locations for PSFUs in the Republic of Serbia. The driving distances between the cities and potential PSFU locations are calculated by using given locations. The largest instance contains all 165 cities and 234 locations, while the smaller-size instances are obtained by grouping cities and locations that belong to one or more neighbor administrative regions in Serbia. The average number of incidents on a monthly basis $f_i$ for a city $i\in I$ is obtained from the data provided by the Statistical Office of the Republic of Serbia, see [33].

In the single-period case ($|T|=1$), the PSORVNS method was tested on instances i12, i6_7_8, i1_2_3_4 and i_all used in [33], and the obtained results were compared with the EA-LS approach proposed in the same paper. In order to benchmark PSO-RVNS for the multi-period case, we have generated instances with $|T|=2$ and $|T|=3$ periods by modifying instances from [33].

In Table 1 we give the overview of test instances used in our computational study. The first two colu-

mns contain the name of the instance and its description, respectively. The name of a single-period instance is the same as in [33], while the name of a multi-period instance includes the number of periods. For example, i12_t2 denotes the instance with two periods that is generated from instance i12 used in [33]. Column headings $|I|$, $|J|$, and $|T|$ in Table 1 represent the number of cities, potential locations and time periods, respectively. The values of $f_{ti}$ are generated in respect to the values of $f_i$ from [33]. For $|T| = 2$, the values of $f_{ti}$ are in the ratio 1:2, while for $|T| = 3$, the corresponding ratio is 1:1:2. As in [33], it is allowed that the values fti increase up to 5% from their nominal values. For each instance presented in Table 1, different values of $k_{max}$, and parameter Γ are considered, while the values of parameters $k_{t,min}$, $t \in T$ are set to 1.

**Table 1.** Overview of test instances used in computational study

| Name | Description | $\|I\|$ | $\|J\|$ | $\|T\|$ | $f_{ti}$ |
|---|---|---|---|---|---|
| i12 | used in Stanimirović et al. (2014) | 17 | 21 | 1 | $f_{1i} = f_i$ |
| i6_7_8 | used in Stanimirović et al. (2014) | 32 | 46 | 1 | $f_{1i} = f_i$ |
| i1_2_3_4 | used in Stanimirović et al. (2014) | 62 | 95 | 1 | $f_{1i} = f_i$ |
| i_all | used in Stanimirović et al. (2014) | 165 | 234 | 1 | $f_{1i} = f_i$ |
| i12_t2 | obtained from i12, $\|T\| = 2$ | 17 | 21 | 2 | $f_{1i} = f_i/3$, $f_{2i} = 2f_i/3$ |
| i6_7_8_t2 | obtained from i6_7_8, $\|T\| = 2$ | 32 | 46 | 2 | $f_{1i} = f_i/3$, $f_{2i} = 2f_i/3$ |
| i1_2_3_4_t2 | obtained from i1_2_3_4, $\|T\| = 2$ | 62 | 95 | 2 | $f_{1i} = f_i/3$, $f_{2i} = 2f_i/3$ |
| i_all_t2 | obtained from i_all, $\|T\| = 2$ | 165 | 234 | 2 | $f_{1i} = f_i/3$, $f_{2i} = 2f_i/3$ |
| i12_t3 | obtained from i12, $\|T\| = 3$ | 17 | 21 | 3 | $f_{1i} = f_{2i} = f_i/4$, $f_{3i} = f_i/2$ |
| i6_7_8_t3 | obtained from i6_7_8, $\|T\| = 3$ | 32 | 46 | 3 | $f_{1i} = f_{2i} = f_i/4$, $f_{3i} = f_i/2$ |
| i1_2_3_4_t3 | obtained from i1_2_3_4, $\|T\| = 3$ | 62 | 95 | 3 | $f_{1i} = f_{2i} = f_i/4$, $f_{3i} = f_i/2$ |
| i_all_t3 | obtained from i_all, $\|T\| = 3$ | 165 | 234 | 3 | $f_{1i} = f_{2i} = f_i/4$, $f_{3i} = f_i/2$ |

**Table 2.** The results of parameter analysis test

| Inst. | Par. | SS | DF | MS | $F$ | $p$ |
|---|---|---|---|---|---|---|
| i12 | $\|N_r\|$ | 0.000 | 1 | 0.000 | 0.0000 | 1.0000 |
| i12 | $c_r$ | 0.000 | 2 | 0.000 | 0.0000 | 1.0000 |
| i12 | $rep$ | 0.000 | 1 | 0.000 | 0.0000 | 1.0000 |
| i12 | $p_{el}$ | 0.000 | 1 | 0.000 | 0.0000 | 1.0000 |
| i6_7_8_t2 | $\|N_r\|$ | 0.133 | 1 | 0.133 | 1.4268 | 0.6008 |
| i6_7_8_t2 | $c_r$ | 1.008 | 2 | 0.504 | 9.0634 | 0.0015 |
| i6_7_8_t2 | $rep$ | 0.246 | 1 | 0.246 | 2.1214 | 0.4879 |
| i6_7_8_t2 | $p_{el}$ | 0.016 | 1 | 0.016 | 0.1992 | 0.9254 |
| i1_2_3_4_t3 | $\|N_r\|$ | 0.000 | 1 | 0.000 | 0.0000 | 1.0000 |
| i1_2_3_4_t3 | $c_r$ | 0.000 | 2 | 0.000 | 0.0000 | 1.0000 |
| i1_2_3_4_t3 | $rep$ | 0.000 | 1 | 0.000 | 0.0000 | 1.0000 |
| i1_2_3_4_t3 | $p_{el}$ | 0.000 | 1 | 0.000 | 0.0000 | 1.0000 |
| i_all | $\|N_r\|$ | 1.464 | 1 | 1.464 | 2.0260 | 0.5013 |
| i_all | $c_r$ | 0.542 | 2 | 0.271 | 0.3387 | 0.7165 |
| i_all | $rep$ | 0.203 | 1 | 0.203 | 0.2599 | 0.9041 |
| i_all | $p_{el}$ | 1.497 | 1 | 1.497 | 2.0763 | 0.4942 |

### 5.1. Calibration of the PSO-RVNS parameters

In order to attain best performance of the proposed PSO-RVNS algorithm, we have performed experimental analysis of several parameters:

- $|N_r|$ – the size of population;
- $c_r$ – coefficient used in acceptance criterion for the newly generated solution in RVNS part;
- $rep$ – the maximal number of subsequent generations without improvement of the best solution;
- $p_{el}$ – the percentage of elite individuals in the population.

A full factorial design experiment is conducted to obtain the best combination of parameters [24]. The levels of parameters used in the experiment are as follows:

- parameter $|N_r|$ has two levels – 30 and 60;
- parameter $c_r$ has three levels – 1, 1.001 and 1.005;
- parameter $rep$ has two levels – 1000 and 2000;
- parameter $p_{el}$ has two levels – 66:67% and 75%.

The total number of combinations is $2 \cdot 3 \cdot 2 \cdot 2 = 24$. In order to test all considered combinations, we use the subset of four problem instances: i12, i6_7_8_t2, i1_2_3_4_t3, and i_all. On each instance,

PSO-RVNS algorithm was run 15 times for each combination of parameters. For each considered instance and each parameter combination, the best objective value obtained by PSORVNS was memorized.

In Table 2, we present the results of the one-way analysis of variance (ANOVA) for the conducted experiments, see [24]. The column headings in Table 2 have the following meaning:

- Inst. – instance's name;
- Par. – parameter that is being tested;
- SS – sum of squares between groups;
- DF – degrees of freedom between group;
- MS – mean of squares between groups;
- $F$ – the value obtained by $F$-test;
- $p$ – corresponding $p$-value.

Critical value used in the parameter analysis test is set to 0.05. According to $p$-values presented in the last column of Table 2, only parameter $c_r$ has some effect on the objective value of the best PSO-RVNS solution (see the results obtained for instances i6_7_8_t2 and i_all). For instances i12 and i1_2_3_4_t3, each combination of parameters leaded to the same result, and therefore no parameter has significant effect when testing these two instances. We may conclude that parameters $|N_r|$, $rep$ and $p_{el}$ do not have significant effect on the obtained PSO-RVNS results. In all cases, the value of $c_r = 1.001$ showed to be the best choice, while the values of other three parameters are set to $|N_r| = 30$, $rep = 1000$, and $p_{el} = 75\%$.

### 5.2. Results and comparisons for single-period case

In this subsection we present the computational results of the proposed PSO-RVNS for single-period case and compare them with the results of EA-LS method from [33] and CPLEX 12.1 commercial solver. The results and comparisons on the set of instances with $|T| = 1$ period, different values of $k_{max}$ and protection parameter Γ are presented in Tables 3 – 7. The meaning of the column headings through these tables are as follows:

- $k_{max}$ – maximal number of locations to be established;
- Γ – the value of parameter controlling the level of robustness;
- Sol. – the objective value of optimal solution (obtained by CPLEX 12.1) or best-known solution (when no optimal solution is known);
- $t_{CPLEX}[s]$ – running time of CPLEX 12.1 solver for the robust model from [33];
- $t_{CPLEX}^{impr}[s]$ – running time of CPLEX 12.1 obtained the robust model from [33] when using Theorem 2;
- EA-LS$_{best}$ – the objective value of the best EA-LS solution, with mark *opt* or *best* when it coincides with the objective value of optimal or best known solution, respectively;
- PSO-RVNS$_{best}$ – the objective value of the best PSO-RVNS solution, with mark *opt* or *best* when it coincides with the objective value of optimal or best known solution, respectively;
- $t_{EA-LS}[s]$ – average time (in seconds) for which EA-LS produced its the best solution;
- $t_{PSO-RVNS}[s]$ – average time (in seconds) for which PSO-RVNS method produced its best solution;
- Incr[%] – increment of the objective value of the solution for the given level of robustness Γ.

In Table 3, the results for instance i12 and $k_{max} \in \{3,4,5\}$ are presented. Both EA-LS and PSO-RVNS reached all optimal results previously obtained by CPLEX 12.1 solver. From the values presented in column *Impr[%]*, it may be noticed that objective function value increases as parameter Γ increases. For example, for $k_{max} = 3$, the increment of the objective function ranges from 0% for $\Gamma = 0$ (no deviation of $f_{ti}$) up to 10.919% for $\Gamma = 17$ (when all values of $f_{ti}$ are changed). Similarly, for $k_{max} \in \{4,5\}$, the objective function value increases from 0% for $\Gamma = 0$ up to 15.232% for $\Gamma = 17$.

**Table 3.** Results and comparisons for instance i12 (single-period case)

| $k_{max}$ | Γ | Sol. | $t_{CPLEX}[s]$ | $t_{CPLEX}^{impr}[s]$ | EA-LS$_{best}$ | PSO-RVNS$_{best}$ | $t_{\text{EA-LS}}[s]$ | $t_{\text{PSO-RVNS}}[s]$ | Incr[%] |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 0 | 27.5500 | 0.37 | 0.37 | *opt* | *opt* | 0.009 | 1.070 | 0.000 |
| 3 | 5 | 29.6708 | 0.76 | 0.37 | *opt* | *opt* | 0.006 | 1.070 | 7.698 |
| 3 | 10 | 30.2250 | 0.91 | 0.37 | *opt* | *opt* | 0.008 | 1.070 | 9.710 |
| 3 | 15 | 30.5458 | 0.48 | 0.37 | *opt* | *opt* | 0.009 | 1.070 | 10.874 |
| 3 | 17 | 30.5583 | 0.50 | 0.37 | *opt* | *opt* | 0.006 | 1.070 | 10.919 |
| 4 | 0 | 19.7500 | 0.34 | 0.34 | *opt* | *opt* | 0.042 | 0.680 | 0.000 |
| 4 | 5 | 21.8708 | 0.49 | 0.34 | *opt* | *opt* | 0.036 | 0.680 | 10.738 |
| 4 | 10 | 22.4250 | 0.51 | 0.34 | *opt* | *opt* | 0.041 | 0.680 | 13.544 |
| 4 | 15 | 22.7458 | 0.68 | 0.34 | *opt* | *opt* | 0.041 | 0.680 | 15.169 |
| 4 | 17 | 22.7583 | 0.77 | 0.34 | *opt* | *opt* | 0.036 | 0.680 | 15.232 |
| 5 | 0 | 19.7500 | 0.16 | 0.16 | *opt* | *opt* | 0.012 | 1.020 | 0.000 |
| 5 | 5 | 21.8708 | 0.44 | 0.16 | *opt* | *opt* | 0.011 | 1.020 | 10.738 |
| 5 | 10 | 22.4250 | 0.42 | 0.16 | *opt* | *opt* | 0.011 | 1.020 | 13.544 |
| 5 | 15 | 22.7458 | 0.50 | 0.16 | *opt* | *opt* | 0.011 | 1.020 | 15.169 |
| 5 | 17 | 22.7583 | 0.59 | 0.16 | *opt* | *opt* | 0.011 | 1.020 | 15.232 |
| Average | | 24.51 | 0.53 | 0.29 | *opt* | *opt* | 0.02 (0.03*) | 0.92 | 9.90 |

**Table 4.** Results and comparisons for instance i6_7_8 (single-period case)

| $k_{max}$ | Γ | Sol. | $t_{CPLEX}[s]$ | $t_{CPLEX}^{impr}[s]$ | EA-LS$_{best}$ | PSO-RVNS$_{best}$ | $t_{\text{EA-LS}}[s]$ | $t_{\text{PSO-RVNS}}[s]$ | Incr[%] |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 0 | 29.1667 | 7.1 | 7.1 | *opt* | *opt* | 0.875 | 1.988 | 0.000 |
| 10 | 10 | 37.0583 | 7.1 | 20.9 | *opt* | *opt* | 0.777 | 1.988 | 27.057 |
| 10 | 20 | 39.3625 | 7.1 | 55.6 | *opt* | *opt* | 1.031 | 1.988 | 34.957 |
| 10 | 25 | 39.9333 | 7.1 | 68.0 | *opt* | *opt* | 1.074 | 1.988 | 36.914 |
| 10 | 32 | 40.2625 | 7.1 | 91.2 | *opt* | *opt* | 1.000 | 1.988 | 38.043 |
| 11 | 0 | 29.1667 | 10.3 | 10.3 | *opt* | *opt* | 0.831 | 2.514 | 0.000 |
| 11 | 10 | 37.0583 | 10.3 | 50.6 | *opt* | *opt* | 1.325 | 2.514 | 27.291 |
| 11 | 20 | 39.3625 | 10.3 | 107.0 | *opt* | *opt* | 1.141 | 2.514 | 35.259 |
| 11 | 25 | 39.9333 | 10.3 | 164.5 | *opt* | *opt* | 1.127 | 2.514 | 37.233 |
| 11 | 32 | 40.2625 | 10.3 | 140.0 | *opt* | *opt* | 1.464 | 2.514 | 38.371 |
| 12 | 0 | 28.4417 | 7.0 | 7.0 | *opt* | *opt* | 1.577 | 1.809 | 0.000 |
| 12 | 10 | 36.3333 | 7.0 | 21.2 | *opt* | *opt* | 1.168 | 1.809 | 27.746 |
| 12 | 20 | 38.6375 | 7.0 | 314.1 | *opt* | *opt* | 1.670 | 1.809 | 35.848 |
| 12 | 25 | 39.2083 | 7.0 | 271.1 | *opt* | *opt* | 1.770 | 1.809 | 37.855 |
| 12 | 32 | 39.5375 | 7.0 | 334.0 | *opt* | *opt* | 1.558 | 1.809 | 39.012 |
| Average | | 36.91 | 110.8 | 8.1 | *opt* | *opt* | 1.23 (1.9*) | 2.10 | 27.71 |

The average CPU time of EA-LS method for instance i12 among all values of $k_{max}$ and Γ was 0.02 s, while the average running time of PSORVNS was 0.923 s. However, it should be mentioned that EA-LS method was tested on a machine with Intel Core i7-860 2.8 GHz processor and 8 GB RAM memory, which has higher performances compared to the computing machine used for experiments in this study (Intel Core i5-2430M 2.4 GHz processor with 8 GB RAM memory). Due to difference in computing platforms, we have normalized the average computational time of the EA-LS by using the approach described in [10] and the data from http://www.cpubenchmark.net/. The average normalized EA-LS running time NAT(EA-LS) is equal to product of average EA-LS time AT(EALS) multiplied by

$$\frac{PCPUS(Intel\ Core\ i7{-}860\ 2.8\ GHz)}{PCPUS(Intel\ Core\ i5{-}2430M\ 2.4\ GHz)}.$$

where PCPUS stands for Passmark CPU Score. Therefore, the calculated normalized average EALS running time for instance i12 is 0.03 seconds, which is given in the last row of Table 3 with mark *.

Note that in [33], the CPLEX 12.5 solver was applied to the robust model involving $|I||J| + |J| + |G| + 2$ constraints and $2|I||J| + |G||J| + |I| + 3|J| + |G| + 4$ variables. From Theorem 2, it follows that solutions for the robust case $\Gamma > 0$ can be easily calculated by using solution that was previously obtained for the deterministic case $\Gamma = 0$. Therefore, in cases when $\Gamma > 0$, we use the results of Theorem 2 and employ CPLEX 12.1 solver on the model with $|I||J| + |J| + 1$ variables and $2|I||J| + |I| + 3|J| + 3$ constraints. In order to investigate the effects of the applied time-saving strategy when solving the robust model, we have tested both variants by using CPLEX 12.1 solver on the same machine – Intel Core i5-2430M 2.4 GHz processor with 8 GB RAM. By comparing CPLEX 12.1 running times presented in columns $t_{CPLEX}[s]$ and $t_{CPLEX}^{impr}[s]$, it can be seen that the use of Theorem 2 speeds up optimization process and enables CPLEX 12.1 to produce optimal solutions in significantly shorter CPU times. The average running time of CPLEX 12.1 was 0.53 seconds when solving the robust formulation from [33], while only 0.29 seconds was needed when using Theorem 2, which is almost 2 times faster. The same strategy was also used in the proposed PSO-RVNS method when calculating objective function values for the robust model. The PSO-RVNS uses the objective value of a solution obtained for the deterministic case ($\Gamma = 0$) to efficiently calculate the corresponding objective value in the robust case ($\Gamma > 0$). For this reason, for each considered instance, computational times of PSO-RVNS obtained for the same value of parameter kmax are the same for all $\Gamma \in \{0, \ldots, |I|\}$.

In Table 4, we present results for instance i6_7_8 and $k_{max} \in \{10,11,12\}$ in the same way as in Table 4. Both EA-LS and PSO-RVNS methods reach optimal solutions, which were previously obtained by CPLEX 12.1 solver. As in the case of instance i12, it may be noticed that objective function value increases as the value of parameter Γ increases. The highest increment of the objective function value is obtained for $\Gamma = 32$: for $k_{max} = 10$, the objective function value is increased by 38.043%, while for $k_{max} = 11$ and $k_{max} = 12$, the objective value is increased by 38.371% and 39.012%, respectively. The average computational time of PSO-RVNS was 2.10 seconds, while the EA-LS showed to be slightly faster, since its normalized average running time was 1.9 seconds. Time savings obtained by using Theorem 2 when solving the robust model by CPLEX 12.1 are more obvious in the case of instance i6_7_8. In average, CPLEX 12.1 needed 110.8 seconds when solving robust model from [33] for instance i6_7_8 among all values of $k_{max}$ and Γ. However, when applying the results of Theorem 2, CPLEX 12.1 needed only 8.1 seconds (in average), which is more than 13 times faster.

In Table 5 we present the results obtained for instance i1_2_3_4 and $k_{max} \in \{21,22,23\}$. The EA-LS and PSO-RVNS methods were successful in reaching

**Table 5.** Results and comparisons for instance i1_2_3_4 (single-period case)

| $k_{max}$ | Γ | Sol. | EA-LS$_{best}$ | PSO-RVNS$_{best}$ | $t_{\text{EA-LS}}[s]$ | $t_{\text{PSO-RVNS}}[s]$ | Incr[%] |
|---|---|---|---|---|---|---|---|
| 21 | 0 | 63.7500 | *opt* | *opt* | 0.764 | 6.923 | 0.000 |
| 21 | 15 | 80.8750 | *opt* | *opt* | 0.725 | 6.923 | 26.863 |
| 21 | 30 | 87.3667 | *opt* | *opt* | 0.634 | 6.923 | 37.046 |
| 21 | 45 | 90.8417 | *opt* | *opt* | 0.673 | 6.923 | 42.497 |
| 21 | 55 | 92.2708 | *opt* | *opt* | 0.736 | 6.923 | 44.738 |
| 21 | 62 | 92.6750 | *opt* | *opt* | 0.862 | 6.923 | 45.372 |
| 22 | 0 | 63.7500 | *opt* | *opt* | 0.502 | 5.144 | 0.000 |
| 22 | 15 | 80.8750 | *opt* | *opt* | 0.564 | 5.144 | 26.863 |
| 22 | 30 | 87.3667 | *opt* | *opt* | 0.474 | 5.144 | 37.046 |
| 22 | 45 | 90.8417 | *opt* | *opt* | 0.624 | 5.144 | 42.497 |
| 22 | 55 | 92.2708 | *opt* | *opt* | 0.584 | 5.144 | 44.738 |
| 22 | 62 | 92.6750 | *opt* | *opt* | 0.625 | 5.144 | 45.372 |
| 23 | 0 | 63.7500 | *opt* | *opt* | 0.502 | 3.238 | 0.000 |
| 23 | 15 | 80.8750 | *opt* | *opt* | 0.434 | 3.238 | 26.863 |
| 23 | 30 | 87.3667 | *opt* | *opt* | 0.764 | 3.238 | 37.046 |
| 23 | 45 | 90.8417 | *opt* | *opt* | 0.622 | 3.238 | 42.497 |
| 23 | 55 | 92.2708 | *opt* | *opt* | 0.389 | 3.238 | 44.738 |
| 23 | 62 | 92.6750 | *opt* | *opt* | 0.712 | 3.238 | 45.372 |
| Average | | 84.63 | *opt* | *opt* | 0.62 (0.9*) | 5.10 | 32.75 |

optimal solutions provided by CPLEX 12.1. From the last column, it may be noticed that objective function value increases as parameter Γ increases. The highest increment of the objective function value is obtained for $\Gamma = 62$. For $k_{max} \in \{21,22,23\}$ objective function value is increased up to 45.37%. The average CPU time of PSO-RVNS method for instance i1_2_3_4 among all values of $k_{max}$ and Γ was 5.1 s, while the normalized average CPU time of EA-LS was 0.9 seconds.

The proposed PSO-RVNS method shows its advantages when the size of instances increases. The results and comparisons for the largest instance i_all is presented in Table 6. We have considered the same values for parameter $k_{max} \in \{4,24,36,48,60\}$ as in [33]. Note that in the case of instance i_all, CPLEX 12.1 solver was unable to provide optimal solutions with the given time limit of 3 h. For $k_{max} = 4$, $k_{max} = 36$, $k_{max} = 48$ and $k_{max} = 60$, both PSO-RVNS and EA-LS obtained the same (best-known) solutions. However, for $k_{max} = 24$, the proposed PSO-RVNS improved the best EA-LS solutions for all $0 \leq \Gamma \leq 165$. The PSO-RVNS was also superior compared to the EA-LS regarding CPU times for the largest instance i_all. The normalized average computational time of EA-LS method for instance i_all among all values of $k_{max}$ and Γ was 79.35 seconds, while the average computational time of the proposed PSO-RVNS was 22.33 seconds, which is around 3.5 shorter. From the last column, it may be noticed that objective function value increases with the increase of protection parameter Γ. The highest increment of the objective function value is obtained for $\Gamma = 165$: for $k_{max} = 4$, $k_{max} = 24$ and $k_{max} \in \{36,48,60\}$, objective function value is increased by 10.985%, 47.395% and 48.354%, respectively.

The presented results for the single-period case show that PSO-RVNS reached all optimal and best-known solutions from [33]. In several cases of the largest considered instance i_all, the proposed PSO-RVNS improved best EA-LS solutions from the same paper. Regarding CPU times, the EA-LS appears to be more efficient than PSORVNS when solving small size instances, while it outperformed EA-LS in the terms of computational time for the largest instance i_all. Computational results obtained with CPLEX 12.1 instance show significant improvements regarding CPU times when using the results of Theorem 2.

### 5.3. Results for $|T| > 1$

In this subsection we present the results on PSO-RVNS instances with $|T| = 2$ and $|T| = 3$ time periods. For each instance, we consider the different values of parameter $k_{max}$. Similarly, as in the case with one period, the objective values in the robust case may be easily calculated by using the corresponding values in the deterministic case. The values of the parameter $k_{max}$ for each instance represent the product of number of periods and value of $k_{max}$ for corresponding instance with one period. For example, for instance i12 with one period, the values of $k_{max}$ are 3, 4 and 5, for twoperiod instance i12_t2 the considered values of $k_{max}$ are 6, 8 and 10, while for three-period instance i12_t3, the values of $k_{max}$ are equal to 9, 12 and 15, etc.

The results of of PSO-RVNS obtained on multi-period instances in the deterministic case ($\Gamma = 0$) are presented in Table 7. This table also contains optimal solution obtained by CPLEX 12.1 solver and the corresponding running time $t_{CPLEX}^{impr}[s]$. In cases when CPLEX found no solution within the given time limit of 3h, mark - is placed the corresponding row. When PSO-RVNS reached optimal solution obtained by CPLEX 12.1, it is denoted by mark *opt*. The remainder of the Table 7 follows the structure of tables from the previous subsection. As it can be seen from Table 7, CPLEX 12.1 provided optimal solutions for 15 out of 28 instances only. The proposed PSORVNS method quickly reached all known optimal solutions, but also provided solutions for 13 instances that remained out of reach of CPLEX 12.1 solver. The

**Table 6.** Results and comparisons for instance i_all and $k_{max} \in \{4,24,36,48,60\}$ (single-period case)

| $k_{max}$ | Γ | Sol. | EA-LS$_{best}$ | PSO-RVNS$_{best}$ | $t_{\text{EA-LS}}[s]$ | $t_{\text{PSO-RVNS}}[s]$ | Incr[%] |
|---|---|---|---|---|---|---|---|
| 4 | 0 | 483.6917 | *best* | *best* | 48.677 | 23.499 | 0.000 |
| 4 | 15 | 504.2542 | *best* | *best* | 52.734 | 23.499 | 4.078 |
| 4 | 30 | 516.1583 | *best* | *best* | 35.366 | 23.499 | 6.290 |
| 4 | 45 | 523.9958 | *best* | *best* | 49.120 | 23.499 | 7.692 |
| 4 | 60 | 528.9667 | *best* | *best* | 36.289 | 23.499 | 8.559 |
| 4 | 75 | 532.7625 | *best* | *best* | 41.764 | 23.499 | 9.211 |
| 4 | 90 | 535.8667 | *best* | *best* | 28.730 | 23.499 | 9.737 |
| 4 | 105 | 538.3542 | *best* | *best* | 48.150 | 23.499 | 10.154 |
| 4 | 120 | 540.4208 | *best* | *best* | 46.912 | 23.499 | 10.497 |
| 4 | 135 | 541.9000 | *best* | *best* | 52.284 | 23.499 | 10.742 |
| 4 | 150 | 542.8958 | *best* | *best* | 34.576 | 23.499 | 10.905 |
| 4 | 165 | 543.3792 | *best* | *best* | 34.972 | 23.499 | 10.985 |
| 24 | 0 | 66.2500 | 76.7500 | *best* | 92.374 | 16.024 | 0.000 |
| 24 | 15 | 86.8125 | 98.9653 | *best* | 78.782 | 16.024 | 23.686 |
| 24 | 30 | 98.7167 | 109.8166 | *best* | 85.067 | 16.024 | 32.889 |
| 24 | 45 | 106.5542 | 118.1625 | *best* | 77.434 | 16.024 | 37.825 |
| 24 | 60 | 111.5250 | 122.5306 | *best* | 83.975 | 16.024 | 40.596 |
| 24 | 75 | 115.3208 | 127.0700 | *best* | 85.394 | 16.024 | 42.552 |
| 24 | 90 | 118.4250 | 128.8611 | *best* | 81.940 | 16.024 | 44.057 |
| 24 | 105 | 120.9125 | 131.7458 | *best* | 99.130 | 16.024 | 45.208 |
| 24 | 120 | 122.9792 | 134.4950 | *best* | 82.663 | 16.024 | 46.129 |
| 24 | 135 | 124.4583 | 136.0944 | *best* | 80.534 | 16.024 | 46.769 |
| 24 | 150 | 125.4542 | 137.6958 | *best* | 509.596 | 16.024 | 47.192 |
| 24 | 165 | 125.9375 | 137.9986 | *best* | 76.599 | 16.024 | 47.395 |
| 36 | 0 | 63.7500 | *best* | *best* | 53.7 | 14.9 | 0.00 |
| 36 | 15 | 84.3125 | *best* | *best* | 34.6 | 14.9 | 24.38 |
| 36 | 30 | 96.2167 | *best* | *best* | 43.8 | 14.9 | 33.74 |
| 36 | 45 | 104.0542 | *best* | *best* | 45.4 | 14.9 | 38.73 |
| 36 | 60 | 109.0250 | *best* | *best* | 37.0 | 14.9 | 41.52 |
| 36 | 75 | 112.8208 | *best* | *best* | 44.8 | 14.9 | 43.49 |
| 36 | 90 | 115.9250 | *best* | *best* | 44.4 | 14.9 | 45.00 |
| 36 | 105 | 118.4125 | *best* | *best* | 37.3 | 14.9 | 46.16 |
| 36 | 120 | 120.4792 | *best* | *best* | 48.6 | 14.9 | 47.08 |
| 36 | 135 | 121.9583 | *best* | *best* | 39.2 | 14.9 | 47.72 |
| 36 | 150 | 122.9542 | *best* | *best* | 51.0 | 14.9 | 48.15 |
| 36 | 165 | 123.4375 | *best* | *best* | 46.0 | 14.9 | 48.35 |
| 48 | 0 | 63.7500 | *best* | *best* | 15.0 | 26.6 | 0.00 |
| 48 | 15 | 84.3125 | *best* | *best* | 13.1 | 26.6 | 24.38 |
| 48 | 30 | 96.2167 | *best* | *best* | 11.6 | 26.6 | 33.74 |
| 48 | 45 | 104.0542 | *best* | *best* | 13.7 | 26.6 | 38.73 |
| 48 | 60 | 109.0250 | *best* | *best* | 11.8 | 26.6 | 41.52 |
| 48 | 75 | 112.8208 | *best* | *best* | 13.1 | 26.6 | 43.49 |
| 48 | 90 | 115.9250 | *best* | *best* | 13.0 | 26.6 | 45.00 |
| 48 | 105 | 118.4125 | *best* | *best* | 12.7 | 26.6 | 46.16 |
| 48 | 120 | 120.4792 | *best* | *best* | 13.7 | 26.6 | 47.08 |
| 48 | 135 | 121.9583 | *best* | *best* | 12.4 | 26.6 | 47.72 |
| 48 | 150 | 122.9542 | *best* | *best* | 11.3 | 26.6 | 48.15 |
| 48 | 165 | 123.4375 | *best* | *best* | 12.8 | 26.6 | 48.35 |
| 60 | 0 | 63.7500 | *best* | *best* | 8.2 | 33.2 | 0.00 |
| 60 | 15 | 84.3125 | *best* | *best* | 7.2 | 33.2 | 24.38 |
| 60 | 30 | 96.2167 | *best* | *best* | 5.4 | 33.2 | 33.74 |
| 60 | 45 | 104.0542 | *best* | *best* | 7.2 | 33.2 | 38.73 |
| 60 | 60 | 109.0250 | *best* | *best* | 7.0 | 33.2 | 41.52 |
| 60 | 75 | 112.8208 | *best* | *best* | 6.1 | 33.2 | 43.49 |
| 60 | 90 | 115.9250 | *best* | *best* | 5.2 | 33.2 | 45.00 |
| 60 | 105 | 118.4125 | *best* | *best* | 5.5 | 33.2 | 46.16 |
| 60 | 120 | 120.4792 | *best* | *best* | 6.8 | 33.2 | 47.08 |
| 60 | 135 | 121.9583 | *best* | *best* | 6.2 | 33.2 | 47.72 |
| 60 | 150 | 122.9542 | *best* | *best* | 7.2 | 33.2 | 48.15 |
| 60 | 165 | 123.4375 | *best* | *best* | 5.6 | 33.2 | 48.35 |
| Average | | 213.39 | 216.24 | *best* (213.39) | 50.98 (79.35*) | 22.33 | 30.87 |

average running time of PSO-RVNS on instances from Table 7 was 53.3 seconds. As in deterministic case, the increase of the objective function values follows the increase of the protection parameter Γ.

Tables 8–9 show the results obtained for modified instance i_all with $|T| = 2$ and $|T| = 3$ periods, respectively. As it was expected, these instances could not be solved to optimality by CPLEX 12.1 solver within the given time limit of 3 hours. For all instances from Tables 8–9 PSORVNS achieved its best solutions in short CPU times. The average running time of PSO-RVNS was 53.2 seconds for i_all_t2 and 129.6 seconds for i_all_t3. The average increase of the objective function value was 62.54% and 85.23%, respectively.

Figure 1 shows the increment of the objective function value as a function of parameter Γ in the case of instance i12_t2. The function value increases for

**Table 7.** Results of CPLEX and PSO-RVNS for instances with $\Gamma = 0, |T| = 2$ and $|T| = 3$ (multi-period case)

| Instance | $k_{max}$ | Sol. | $t^{impr}_{\text{CPLEX}}[s]$ | PSO-RVNS$_{best}$ | $t_{\text{PSO-RVNS}}[s]$ |
|---|---|---|---|---|---|
| i12_t2 | 6 | 15.3981 | 6.1 | *opt* | 2.9 |
| i12_t2 | 8 | 13.1667 | 3.6 | *opt* | 3.1 |
| i12_t2 | 10 | 13.1667 | 1.7 | *opt* | 1.8 |
| i6_7_8_t2 | 20 | 18.5000 | 308.5 | *opt* | 1.3 |
| i6_7_8_t2 | 22 | 18.5000 | 180.4 | *opt* | 3.4 |
| i6_7_8_t2 | 24 | 18.5000 | 133.0 | *opt* | 3.4 |
| i1_2_3_4_t2 | 42 | 42.5000 | 1872.2 | *opt* | 9.7 |
| i1_2_3_4_t2 | 44 | 42.5000 | 3426.2 | *opt* | 14.9 |
| i1_2_3_4_t2 | 46 | 42.5000 | 2696.2 | *opt* | 15.3 |
| i_all_t2 | 8 | 258.0741 | – | *best* | 32.7 |
| i_all_t2 | 48 | 42.5000 | – | *best* | 73.6 |
| i_all_t2 | 72 | 42.5000 | – | *best* | 83.9 |
| i_all_t2 | 96 | 42.5000 | – | *best* | 96.0 |
| i_all_t2 | 120 | 42.5000 | – | *best* | 103.5 |
| i12_t3 | 9 | 11.5486 | 78.6 | *opt* | 2.3 |
| i12_t3 | 12 | 9.8750 | 9.8 | *opt* | 5.0 |
| i12_t3 | 15 | 9.8750 | 6.3 | *opt* | 4.5 |
| i6_7_8_t3 | 30 | 13.8750 | 1229.5 | *opt* | 5.5 |
| i6_7_8_t3 | 33 | 13.8750 | 1289.3 | *opt* | 5.5 |
| i6_7_8_t3 | 36 | 13.8750 | 952.6 | *opt* | 4.8 |
| i1_2_3_4_t3 | 63 | 31.8750 | – | *best* | 44.5 |
| i1_2_3_4_t3 | 66 | 31.8750 | – | *best* | 49.0 |
| i1_2_3_4_t3 | 69 | 31.8750 | – | *best* | 69.1 |
| i_all_t3 | 12 | 165.4549 | – | *best* | 100.2 |
| i_all_t3 | 72 | 31.8750 | – | *best* | 159.1 |
| i_all_t3 | 108 | 31.8750 | – | *best* | 204.5 |
| i_all_t3 | 144 | 31.8750 | – | *best* | 206.7 |
| i_all_t3 | 180 | 31.8750 | – | *best* | 187.2 |
| Average | | 39.7968 | – | *best* | 53.3 |

**Table 8.** Results of PSO-RVNS for instance i_all_t2 and $|T| = 2$ (multi-period case)

| $k_{max}$ | $\Gamma$ | PSO-RVNS$_{best}$ | $t_{\text{PSO-RVNS}}[s]$ | Incr[%] |
|---|---|---|---|---|
| 8 | 0 | 258.0741 | 32.7 | 0.00 |
| 8 | 20 | 275.4060 | 32.7 | 6.72 |
| 8 | 40 | 284.7963 | 32.7 | 10.35 |
| 8 | 60 | 291.8616 | 32.7 | 13.09 |
| 8 | 80 | 297.0685 | 32.7 | 15.11 |
| 8 | 100 | 300.9907 | 32.7 | 16.63 |
| 8 | 120 | 304.2269 | 32.7 | 17.88 |
| 8 | 140 | 306.9532 | 32.7 | 18.94 |
| 8 | 160 | 309.1213 | 32.7 | 19.78 |
| 8 | 180 | 311.0130 | 32.7 | 20.51 |
| 8 | 200 | 312.6352 | 32.7 | 21.14 |
| 8 | 220 | 313.9880 | 32.7 | 21.67 |
| 8 | 240 | 315.1019 | 32.7 | 22.10 |
| 8 | 260 | 316.0546 | 32.7 | 22.47 |
| 8 | 280 | 316.8088 | 32.7 | 22.76 |
| 8 | 300 | 317.3463 | 32.7 | 22.97 |
| 8 | 320 | 317.6977 | 32.7 | 23.10 |
| 8 | 330 | 317.7616 | 32.7 | 23.13 |
| 48 | 0 | 42.5000 | 73.6 | 0.00 |
| 48 | 20 | 59.8319 | 73.6 | 40.78 |
| 48 | 40 | 69.2222 | 73.6 | 62.88 |
| 48 | 60 | 76.2875 | 73.6 | 79.50 |
| 48 | 80 | 81.4944 | 73.6 | 91.75 |
| 48 | 100 | 85.4167 | 73.6 | 100.98 |
| 48 | 120 | 88.6528 | 73.6 | 108.59 |
| 48 | 140 | 91.3792 | 73.6 | 115.01 |
| 48 | 160 | 93.5472 | 73.6 | 120.11 |
| 48 | 180 | 95.4389 | 73.6 | 124.56 |
| 48 | 200 | 97.0611 | 73.6 | 128.38 |
| 48 | 220 | 98.4139 | 73.6 | 131.56 |
| 48 | 240 | 99.5278 | 73.6 | 134.18 |
| 48 | 260 | 100.4806 | 73.6 | 136.42 |
| 48 | 280 | 101.2347 | 73.6 | 138.20 |
| 48 | 300 | 101.7722 | 73.6 | 139.46 |
| 48 | 320 | 102.1236 | 73.6 | 140.29 |
| 48 | 330 | 102.1875 | 73.6 | 140.44 |
| Average | | 195.9299 | 53.2 | 62.54 |

**Table 9.** Results of PSO-RVNS for instance i_all_t3 and $|T| = 3$ (multi-period case)

| $k_{max}$ | $\Gamma$ | PSO-RVNS$_{best}$ | $t_{\text{PSO-RVNS}}[s]$ | Incr[%] |
|---|---|---|---|---|
| 12 | 0 | 165.4549 | 100.2 | 0.00 |
| 12 | 25 | 180.9153 | 100.2 | 9.34 |
| 12 | 50 | 189.2726 | 100.2 | 14.40 |
| 12 | 75 | 195.9538 | 100.2 | 18.43 |
| 12 | 100 | 201.1715 | 100.2 | 21.59 |
| 12 | 125 | 205.1184 | 100.2 | 23.97 |
| 12 | 150 | 208.3965 | 100.2 | 25.95 |
| 12 | 175 | 211.2465 | 100.2 | 27.68 |
| 12 | 200 | 213.5236 | 100.2 | 29.05 |
| 12 | 225 | 215.4361 | 100.2 | 30.21 |
| 12 | 250 | 217.1340 | 100.2 | 31.23 |
| 12 | 275 | 218.6444 | 100.2 | 32.15 |
| 12 | 300 | 219.9674 | 100.2 | 32.95 |
| 12 | 325 | 221.0830 | 100.2 | 33.62 |
| 12 | 350 | 222.0799 | 100.2 | 34.22 |
| 12 | 375 | 222.9632 | 100.2 | 34.76 |
| 12 | 400 | 223.7028 | 100.2 | 35.20 |
| 12 | 425 | 224.2757 | 100.2 | 35.55 |
| 12 | 450 | 224.7194 | 100.2 | 35.82 |
| 12 | 475 | 225.0309 | 100.2 | 36.01 |
| 12 | 495 | 225.1424 | 100.2 | 36.07 |
| 72 | 0 | 31.8750 | 159.1 | 0.00 |
| 72 | 25 | 47.3354 | 159.1 | 48.50 |
| 72 | 50 | 55.6927 | 159.1 | 74.72 |
| 72 | 75 | 62.3740 | 159.1 | 95.68 |
| 72 | 100 | 67.5917 | 159.1 | 112.05 |
| 72 | 125 | 71.5385 | 159.1 | 124.43 |
| 72 | 150 | 74.8167 | 159.1 | 134.72 |
| 72 | 175 | 77.6667 | 159.1 | 143.66 |
| 72 | 200 | 79.9437 | 159.1 | 150.80 |
| 72 | 225 | 81.8562 | 159.1 | 156.80 |
| 72 | 250 | 83.5542 | 159.1 | 162.13 |
| 72 | 275 | 85.0646 | 159.1 | 166.87 |
| 72 | 300 | 86.3875 | 159.1 | 171.02 |
| 72 | 325 | 87.5031 | 159.1 | 174.52 |
| 72 | 350 | 88.5000 | 159.1 | 177.65 |
| 72 | 375 | 89.3833 | 159.1 | 180.42 |
| 72 | 400 | 90.1229 | 159.1 | 182.74 |
| 72 | 425 | 90.6958 | 159.1 | 184.54 |
| 72 | 450 | 91.1396 | 159.1 | 185.93 |
| 72 | 475 | 91.4510 | 159.1 | 186.91 |
| 72 | 495 | 91.5625 | 159.1 | 187.25 |
| Average | | 144.2211 | 129.6 | 85.23 |

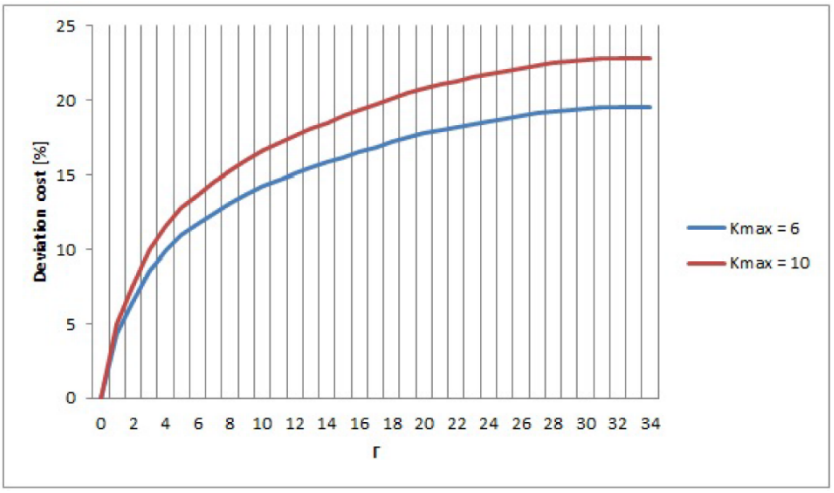


**Figure 1.** Change in the objective function value as a function of Γ for instance i12_t2

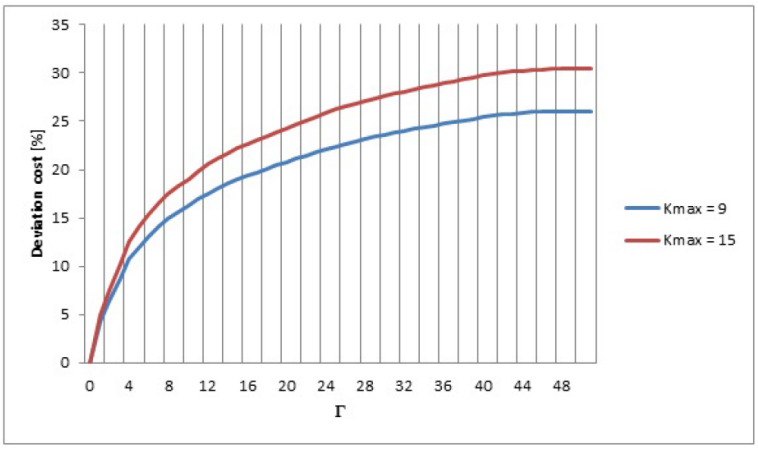


**Figure 2.** Change in the objective function value as a function of Γ for instance i12_t3

$0 \leq \Gamma \leq 34$, while for $\Gamma \geq 34$ it remains unchanged. We present the result graphically for two values of parameter $k_{max}$: 6 and 10. For $k_{max} = 6$, the highest increment is obtained for $\Gamma \geq 34$ and it is equal to 19.537. The value for $k_{max} = 10$ is equal to 22.848. The blue line shows the values for $k_{max} = 6$, and the red one for $k_{max} = 10$.

Similarly, Figure 2 presents the increment of the objective function value as a function of parameter Γ in the case of instance i12_t3. The function value increases for $0 \leq \Gamma \leq 51$, while for $\Gamma \geq 51$ it remains unchanged. We present the result graphically for two different values of pa rameter $k_{max}$: 9 (blue line) and 15 (red line). For $k_{max} = 9$, the highest increment is obtained for $\Gamma \geq 51$ and it is equal to 26.049. The value for $k_{max} = 15$ is equal to 30.464.

## 6. Conclusions

This study introduces a generalization of the problem of emergency service location from [33]. Having in mind that needs for emergency service may vary on weekly or daily basis, we involve multiple periods in the model proposed in [33]. In addition, we impose lower bounds on the number of services to be located in each period and the upper limit on the total number of available emergency services through all periods. Considering the nature of the problem, we further propose a robust optimization model of the multi-period problem which captures the uncertainty of emergency incidents. It is assumed that input data representing the number of incidents in the considered city and time period are subject to uncertainty, and they are modeled as independent and bounded random variables with unknown distribution.

Both deterministic and robust variant of the multi-period model are tested by CPLEX 12.1 solver on the set of modified real-life instances [33]. In the case of robust variant, we propose the strategy that speeds up objective function calculation and therefore, significantly reduces the running time of CPLEX 12.1 solver. In spite of the applied time saving strategy, the largest problem instances remained out of reach for CPLEX within the given time limit of 3h. Therefore, a hybrid optimization approach (PSO-RVNS), based on combination of Particle Swarm Optimization (PSO) and Reduced Variable Neighbourhood Search (RVNS), is proposed. The elements of the proposed hybrid PSO-RVNS are designed for the problem under consideration and its parameters are experimentally adjusted to obtain the best algorithm's performance.

The results of the conducted computational experiments showed that the proposed PSO-RVNS quickly reached all optimal solutions obtained by CPLEX 12.1 for both deterministic and robust variants of the problem. In cases when optimal solutions are known, the PSO-RVNS steadily converged to its best solution in short CPU times. In the single-period case, it has been shown that for largest problem instances, the PSO-RVNS outperformed the EA-LS method from [33] in the case of both solution quality and CPU times. The analysis of the solutions obtained in the robust case of the multi-period problem showed that the value of objective function increases as the protection level parameter Γ increases, as in the single-period case.

Based on the presented results of computational experiments, we conclude that the proposed PSO-RVNS showed to be successful when solving both deterministic and robust variant of the multi-period emergency service location problem, and it represents a promising approach that may be applied to similar location problems. Some directions for future work involve parallelization of the proposed PSO-RVNS method and its hybridization with other heuristic or exact optimization techniques.

## Acknowledgement

This research was partially supported by Serbian Ministry of Education, Science and Technological Development under the grants no. 174010, 47017, and 044006.

## References

[1] **O. Baron, O. Berman, D. Krass, Q. Wang.** The equitable location problem on the plane. *European Journal of Operational Research*, 2007, Vol. 183, No. 2, 578–590.

[2] **A. Ben-Tal, A. Nemirovski.** Robust solutions to uncertain programs. *Operations Research Letters*, 1999, Vol. 25, No. 1,1–13.

[3] **A. Ben-Tal, A. Nemirovski.** Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 2000, Vol. 88, 411–424.

[4] **D. Bertsimas, M. Sim.** Robust discrete optimization and network flows. *Mathematical Programming*, 2003, Vol. 98, 49–71.

[5] **M. Blais, S.D. Lapierre, G. Laporte.** Solving a home-care districting problem in an urban setting. Journal of the *Operational Research Society*, 2003, Vol. 54, 1141–1147.

[6] **C. Blum, J. Puchingerb, G.R. Raidl, A. Roli.** Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 2011, Vol. 11, 4135–4151.

[7] **B. Boffey, R. Galvão, L. Espejo.** A review of congestion models in the location of facilities with immobile servers. *European Journal of Operational Research*, 2007, Vol. 178, No. 3, 643–662.

[8] **L. Brotcone, G. Laporte, F. Semet.** Ambulance location and relocation models. *European Journal of Operational Research*, 2003, Vol. 147, 451–463.

[9] **M. S. Canbolat, M. von Massow.** Locating emergency facilities with random demand for risk minimization. *Expert Systems with Applications*, 2011, Vol. 38, No. 8, 10099–10106.

[10] **J. J. Dongarra.** Performance of Various Computers Using Standard Linear Equations Software. *CS-89-85s, University of Manchester*, 2014.

[11] **J. B. Goldberg.** Operations research models for the deployment of emergency services vehicles. *EMS Management Journal*, 2004, Vol. 1, 20–39.

[12] **P. Hansen, N. Mladenović.** Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 2001, Vol. 130, 449–467.

[13] **J. Kalcsics, S. Nickel, M. Schöder.** Towards a unified territory design approach – Applications, algorithms and GIS integration. *Top*, 2005, Vol. 13, 1– 56.

[14] **J. Kennedy, R. Eberhart.** Particle Swarm Optimization. In: *Proceedings of IEEE International Conference on Neural Networks*, 1995, Vol. 4, 1942– 1948.

[15] **J. Kennedy.** The particle swarm: social adaptation of knowledge. In: *Proceedings of IEEE International Conference on Evolutionary Computation*, 1997, 303–308.

[16] **D. G. Kim, Y. D. Kim.** A branch and bound algorithm for determining locations of long-term care facilities, *European Journal of Operational Research*, 2010, Vol. 206, 168–177.

[17] **R. C. Larson.** A hypercube queueing model for facility location and redistricting in urban emergency service. *Computers and Operations Research*, 1974, Vol. 1, 67–95.

[18] **F. V. Louveaux.** Discrete stochastic location models. *Annals of Operations Research*, 1986, Vol. 6, 87– 94.

[19] **V. Marianov, C. ReVelle.** Siting of emergency services. In: *Z. Drezner (ed). Facility Location: A Survey of Applications and Methods. Springer Verlag, New York*, 1995, 199–223.

[20] **M. Marić, Z. Stanimirović, S. Božović.** Hybrid metaheuristic method for determining locations for long-term health care facilities. *Annals of Operations Research*, 2015, Vol. 227, No. 1, 3–23.

[21] **A. Marin.** The discrete facility location problem with balanced allocation of customers. *European Journal of Operational Research*, 2011, 210(1), 27–38.

[22] **S. Mišković, Z. Stanimirović.** A Memetic Algorithm for Solving Two Variants of the Two-Stage Uncapacitated Facility Location Problem. *Information Technology and Control*, 2013, Vol. 42, No. 2, 1178–190.

[23] **N. Mladenović, P. Hansen.** Variable neighborhood search. *Computers and Operations Research*, 1997, Vol. 24, 1097–1100.

[24] **D. C. Montgomery.** *Design and Analysis of Experiments*. 7th edition, New York: John Wiley & Sons.

[25] **F. Neri, C. Cotta.** Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2012, Vol. 2, 1–14.

[26] **W. Ogryczak.** Inequality measures and equitable approaches to location problems. *European Journal of Operational Research*, 2000, Vol. 122, 374–39.

[27] **L. Özdamar, E. Ekinci, B. Küçükyazici.** Emergency logistics planning in natural disasters. *Annals of Operations Research*, 2004, 129, 217– 245.

[28] **G. Palubeckis, D. Rubliauskas, A. Targamadzė** Metaheuristic approaches for the quadratic minimum spanning tree problem. *Information Technology and Control*, 2010, Vol. 39, No. 4, 257–268.

[29] **R. Poli.** Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008, 685175, 1–10.

[30] **Y. B. Shin, E. Kita.** Search performance improvement of Particle Swarm Optimization by second best particle information. *Applied Mathematics and Computation*, 2014, Vol. 246, 346–354.

[31] **L. V. Snyder.** Facility location under uncertainty: a review. *IIE Transactions*, 2006, Vol. 38, No. 7, 547–564.

[32] **A. L. Soyster.** Convex programming with setinclusive constraints and applications to inexact linear programming. *Operations Research*, 1973, Vol. 21, 1154–1157.

[33] **Z. Stanimirović, I. Grujičić, D. Trifunović.** Modeling the Emergency Service Network of Police Special Forces Units for High-Risk Law Enforcement Operations. *INFOR: Information Systems and Operational Research*, 2014, Vol. 52, No. 4, 206–226.

[34] **A. J. Swersey.** The deployment of Police, Fire and Emergency Medical Units. In: *S.M. Pollock et al. (eds) Handbooks in Operations Rresearch and Management Science. Elsevier Science*, Chapter 6, 151–200.

[35] **E. G. Talbi.** Metaheuristics: From Design to Implementation. *Hoboken, New Jersey: Wiley & Sons*, 2009, 385–454.

[36] **X. Xu, Y. Qi, Z. Hua.** Forecasting demand of commodities after natural disasters. *Expert Systems with Applications*, 2010, Vol. 37, No. 6, 431–437.

Received January 2016.

## Appendix A. Objective function calculation for the case $\Gamma > 0$

For the objective function calculation in the case $\Gamma > 0$, we use the results of Theorem 2.

Proof. (Theorem 2) Let us first assume that $z \neq f_k$ holds for all $k$, $1 \leq k \leq n+1$. Therefore, there exists an index $j$ such that $z = f_j - t$, where $t > 0$. Let $z' = f_j = z + t$ and $r_i' = r_i - t$, $1 \leq i \leq n$. We now have

$$z' + r_i' = (z' - t) + (r_i' + t) = z + r_i \geq f_i, \quad \text{(A.1)}$$

and

$$\begin{aligned} F_\Gamma(z, r_1, \dots, r_n) &= \Gamma z + \sum_{i=1}^{n} r_i \\ &= \Gamma(z' - t) + \sum_{i=1}^{n} (r_i' + t) \\ &= \Gamma z' + \sum_{i=1}^{n} r_i' + (n - \Gamma)t \\ &= F_\Gamma(z', r_1', \dots, r_n') + (n - \Gamma)t. \quad \text{(A.2)} \end{aligned}$$

Since $0 \leq \Gamma \leq n$, it follows that

$$F_\Gamma(z, r_1, \dots, r_n) \geq F_\Gamma(z', r_1', \dots, r_n').$$

We may now conclude that it is enough to consider the case when $z = f_k$ for $k \in \{1, \dots, n+1\}$. Let $z = f_k$ for some $k \in \{1, \dots, n+1\}$.

In order to achieve the minimum of

$$F_\Gamma(z, r_1, \dots, r_n),$$

the following conditions need to be satisfied for all $1 \leq \mathrm{i} \leq n$:

$$r_i = \begin{cases} f_i - z, & \text{if } f_i \geq z, \\ 0, & \text{if } f_i < z. \end{cases} \quad \text{(a.3)}$$

From (A.3) it follows that

$$r_i = \begin{cases} f_i - z, & \text{if } i \leq \mathrm{k}, \\ 0, & \text{if } \mathrm{i} > \mathrm{k}. \end{cases} \quad \text{(a.4)}$$

We now rewrite $F_\Gamma(z, r_1, \dots, r_n)$ as

$$\begin{aligned} F_\Gamma(z, r_1, \dots, r_n) &= \sum_{i=1}^{n} r_i + \Gamma z \\ &= \sum_{i=1}^{k} (f_i - f_k) + \Gamma f_k \\ &= \sum_{i=1}^{k} f_i + (\Gamma - k) f_k. \quad \text{(A.5)} \end{aligned}$$

The following cases may be distinguished:

1° If $k = \Gamma + 1$, then

$$\begin{aligned} \sum_{i=1}^{k} f_i + (\Gamma - k) f_k &= \sum_{i=1}^{\Gamma+1} f_i + (\Gamma - \Gamma - 1) f_{\Gamma+1} \\ &= \sum_{i=1}^{\Gamma} f_i + f_{\Gamma+1} - f_{\Gamma+1} \\ &= \sum_{i=1}^{\Gamma} f_i. \quad \text{(A.6)} \end{aligned}$$

2° If $k > \Gamma + 1$, then

$$\begin{aligned} \sum_{i=1}^{k} f_i + (\Gamma - k) f_k &= \sum_{i=1}^{\Gamma} f_i \\ &\quad + \sum_{i=\Gamma+1}^{\mathrm{k}} f_i + (\Gamma - k) f_k \\ &= \sum_{i=1}^{\Gamma} f_i + \sum_{i=\Gamma+1}^{k} (f_i - f_k). \quad \text{(A.7)} \end{aligned}$$

Since $f_i \geq f_k$ for all $i \in \{\Gamma + 1, \dots, k\}$, for $k > \Gamma + 1$ we have

$$\sum_{i=1}^{\Gamma} f_i + \sum_{i=\Gamma+1}^{k} (f_i - f_k) \geq \sum_{i=1}^{\Gamma} f_i. \quad \text{(A.8)}$$

3° If $k < \Gamma + 1$, then

$$\begin{aligned} &\sum_{i=1}^{\mathrm{k}} f_i + (\Gamma - k) f_k = \sum_{i=1}^{\Gamma} f_i \\ &- \sum_{i=\mathrm{k}+1}^{\Gamma} f_i + (\Gamma - k) f_k = \\ &\sum_{i=1}^{\Gamma} f_i + \sum_{i=\mathrm{k}+1}^{\Gamma} (f_k - f_i). \quad \text{(A.9)} \end{aligned}$$

Note that in this case for all $i \in \{k+1, \dots, \Gamma\}$, we have $f_k \geq f_i$. Therefore, for $k < \Gamma + 1$, we have

$$\sum_{i=1}^{\Gamma} f_i + \sum_{i=k+1}^{\Gamma} (f_k - f_i) \geq \sum_{i=1}^{\Gamma} f_i \quad \text{(A.10)}$$

From cases $1° - 3°$, we may conclude that the minimal value of $F_\Gamma(z, r_1, \dots, r_n) = \Gamma_z + \sum_{i=1}^{n} r_i$ is obtained for $z = f_{\Gamma+1}$, and it is equal to $F_\Gamma^{min} = \sum_{i=1}^{\Gamma} f_i$. □