Metadata of the chapter that will be visualized in SpringerLink

Book Title	Discrete Optimization and Operations Research	
Series Title		
Chapter Title	Variable Neighborhood Search-Based Heuristics for Min-Power Symmetric Connectivity Problem in Wireless Networks	
Copyright Year	2016	
Copyright HolderName	Springer International Publishing Switzerland	
Corresponding Author	Family Name	**Plotnikov**
	Particle	
	Given Name	**Roman**
	Prefix	
	Suffix	
	Division	
	Organization	Sobolev Institute of Mathematics
	Address	Novosibirsk, Russia
	Email	nomad87@ngs.ru
Author	Family Name	**Erzin**
	Particle	
	Given Name	**Adil**
	Prefix	
	Suffix	
	Division	
	Organization	Sobolev Institute of Mathematics
	Address	Novosibirsk, Russia
	Division	
	Organization	Novosibirsk State University
	Address	Novosibirsk, Russia
	Email	
Author	Family Name	**Mladenovic**
	Particle	
	Given Name	**Nenad**
	Prefix	
	Suffix	
	Division	
	Organization	University of Valenciennes and Hainaut-Cambresis
	Address	Famars, France
	Email	

Abstract

We investigate the well-known NP-hard problem of finding an optimal communication subgraph in a given edge-weighted graph. This problem appears in different distributed wireless communication networks, e.g., in wireless sensor networks, when it is necessary to minimize transmission energy consumption. We propose new heuristic algorithms based on variable neighborhood search metaheuristic. Our results have

	been compared with the best known results, and the numerical experiment showed that, on a large number of instances, our algorithms outperform the previous ones, especially in a case of large dimensions.
Keywords (separated by '-')	Wireless sensor networks - Energy efficiency - NP-hard problem - Variable neighborhood search

Variable Neighborhood Search-Based Heuristics for Min-Power Symmetric Connectivity Problem in Wireless Networks

Roman Plotnikov[1](✉), Adil Erzin[1,2], and Nenad Mladenovic[3]

[1] Sobolev Institute of Mathematics, Novosibirsk, Russia
nomad87@ngs.ru
[2] Novosibirsk State University, Novosibirsk, Russia
[3] University of Valenciennes and Hainaut-Cambresis, Famars, France

Abstract. We investigate the well-known NP-hard problem of finding an optimal communication subgraph in a given edge-weighted graph. This problem appears in different distributed wireless communication networks, e.g., in wireless sensor networks, when it is necessary to minimize transmission energy consumption. We propose new heuristic algorithms based on variable neighborhood search metaheuristic. Our results have been compared with the best known results, and the numerical experiment showed that, on a large number of instances, our algorithms outperform the previous ones, especially in a case of large dimensions.

Keywords: Wireless sensor networks · Energy efficiency · NP-hard problem · Variable neighborhood search

AQ1

1 Introduction

In recent years different issues related to the wireless communication networks have been actively researched (see, e.g., [1,16]). Mainly, the problem is to minimize energy consumption of network elements per time unit in order to prolong the lifetime of the network. Since often the exact positions of the network elements and the topology of the network cannot be predefined, modern sensors have ability to adjust their transmission ranges in order to minimize energy consumption without breaking the connectivity of the network. Herewith, usually the energy consumption of a network's element is assumed to be proportional to d^s, where $s \geq 2$ and d is the transmission range [15]. But in the general case this condition may not be satisfied because of the inhomogeneity of the environment, radio interference and peculiar properties of network elements (e.g., the signal may not be spread equally in all directions). Thus, the communication energy consumption for each connection could be arbitrary.

We assume that the communication network is represented as a connected graph $G = (V, E)$. In this paper we consider the symmetric case: an edge between two vertices means that the both of them can send a message to each other and the energy consumption for this communication is the same for both of them. If $c_{ij} \geq 0$ is a transmission-related energy consumption needed for sending data from $i \in V$ to $j \in V$ (as well as from j to i), then in the connected subgraph $T = (V, E'),\ E' \subseteq E$ the energy consumption of node $i \in V$ equals to $E_i(T) = \max\limits_{j:(i,j)\in E'} c_{ij}$. The goal of this paper is the development of algorithms for the construction of a spanning subgraph T that minimizes $\sum\limits_{i\in V} E_i(T)$. Without loss of generality, we assume that subgraph T is a spanning tree.

In this paper we propose new heuristic algorithms which use the variable neighborhood search (VNS) metaheuristic, different local searches and two variants of shaking algorithm. We compare solutions obtained by these algorithms with optimal solutions in small dimension cases and with solutions obtained by other algorithms when dimension is large.

The rest of the paper is organized as follows. Section 2 contains the formulation of the problem. In Sect. 3 the related papers are described. The new heuristics are proposed in Sect. 4. In Sect. 5 the results of numerical experiments are presented. Section 6 concludes the paper.

2 Problem Statement

Mathematically, the considered problem can be formulated as follows. Given a simple connected weighted graph $G = (V, E)$ with a vertex set V, $|V| = n$, and an edge set E, find such spanning tree T^* of G, which is the solution to the following problem:

$$W(T) = \sum_{i\in V} \max_{j\in V_i(T)} c_{ij} \to \min_{T}, \tag{1}$$

where $V_i(T)$ is the set of vertices adjacent to the vertex i in the tree T and $c_{ij} \geq 0$ is the weight of the edge $(i, j) \in E$.

In literature, this problem is called the Minimum Power Symmetric Connectivity Problem (MPSCP) [6]. Any feasible solution of (1), i.e., a spanning tree of G, will be called a *communication tree* (subgraph). It is known that (1) is strongly NP-hard [1,3,4,12], and if P $\neq$ NP, then the problem is inapproximable within $1 + \frac{1}{260}$ [4]. Therefore, the construction and analysis of efficient approximation algorithms are some of the most important issues regarding the research on this problem.

3 Related Works

The more general Range Assignment Problem, where the goal is to find a strong connected subgraph in a given oriented graph, has been considered in [5,12]. Its subproblem, MPSCP, was first studied in [6]. The authors proved that Minimum Spanning Tree (MST) is 2-approximation for this problem. Also they proposed a polynomial-time approximation scheme with performance ratio of $1 + \ln 2 + \varepsilon \approx 1.69$ and 15/8-approximate polynomial algorithm. In [7] a greedy heuristic, later called Incremental Power: Prim (IPP), was proposed. IPP is similar to the Prim's algorithm of finding of MST. A Kruscal-like heuristic, later called Incremental Power: Kruscal, was studied in [8]. Both of these so called incremental power heuristics have been proposed for the Minimum Power Asymmetric Broadcast Problem, but they are suitable for MPSCP too. It is proved in [14] that they both have an approximation ratio 2, and it was shown in the same paper that in practice they yield significantly more accurate solution than MST.

Authors of [1] proposed two local search heuristics. The first one is edge-switching (ES) which iteratively performs the best possible replacement of a tree edge and a non-tree edge until a local optimum is reached. The other local search algorithm is edge and fork switching (EFS), where at each step an attempt to replace one or two edges of a tree by an edge or a fork (two adjacent edges) in the best way. In [14] two ES-like heuristics were proposed. In the first one, ES1a, each non-tree edge is added at first and then an edge which belongs to the appeared cycle and causes the maximum power costs is removed. In the second heuristic, ES1b, each edge from a tree is considered to be replaced by the non-tree edge in such way that decrease of objective is maximum. It should be noticed that instead of finding a local optimum ES1a and ES1b perform a single loop on a fixed list of edges (i.e., once added or removed edges are never considered again). Also, they propose a faster sweep method (SW) and the most time-consuming double edge switching (ES2), which is said to be the generalization of EFS: it performs replacements of two edges from a tree and two non-tree edges while it leads to reduction of the objective. Their numerical experiments demonstrate the weakest results of SW (4–5 % improvement over MST for 50–100 nodes), better results of ES1a and ES1b (about 5.5 %), and incredibly high results of ES2 (12–14 %). However, we, as well as authors of [17], could not achieve even 7 % improvement over MST for these dimensions on random instances using our algorithms. It seems like the optimal solution, on average, does not outperform MST by more than 7 % on our random instances. Anyway, ES2 is not applicable for large dimensions because of the high time complexity ($O(|V|^3|E|^2)$). Also, the another two local searches should be mentioned: ST from [17] and LI from [3]. They are very similar because they use the same idea: at each step an edge is removed from a tree and the root of obtained subtree is reconnected with some vertex from another subtree in such way that the decrease of the objective is maximum. The difference between ST and LI is the following: in ST the best replacement is performed at each step, but in LI all edges are sequently considered to be removed and replaced by another edge in the best way, and this loop is repeated while the solution is improved at least at one its iteration.

In [13] a way to filter the edges without impairment of the optimal solution was proposed. This method allows to significantly simplify the initial communication graph and to reduce the computation time. Authors of [17] presented a new iterated local search (ILS) which uses ES and EFS at local search phase, filtration technique from [13] and two different mutation operators. Their numerical experiment results demonstrate that, on average, the best solution within acceptable time can be obtained by ILS with ES, filtration and so-called random increase mutation.

In [2] a hybrid genetic algorithm (GA), which uses variable neighborhood descent (VND) as mutation, was proposed. This algorithm is well parallelized and very fast.

Since we don't know any better heuristics proposed by other authors, we have implemented the best variant of ILS, hybrid GA with VND and compared our algorithms with them in Sect. 5.

4 Heuristics

As mentioned earlier, we use the VNS metaheuristic idea to get an approximate solution of (1). We use two well-known schemas: basic VNS and general VNS. Detailed descriptions of both these methods can be found in [9,10]. For the reader's convenience the pseudo codes of these algorithms are presented in Figs. 1 and 2. These metaheuristics consist of the local search and shaking phases. As a stopping criteria of VNS-based algorithms we used the following rule: if there were no any improvements in last 3 iterations then algorithm stops.

```
1: Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, ..., k_{\max}$ that will be used
   for the shaking phase, and let $\mathcal{N}_0 = \{x\}$; find an initial solution $x$; set $k = 0$;
2: while the stopping criteria is not met do
3:   while $k \le k_{\max}$ do
4:     Perform Shaking: generate a point $x'$ at random from $\mathcal{N}_k(x)$;
5:     Perform a Local search. Let $x''$ be an obtained local optimum;
6:     if $x''$ is better than $x$ then
7:       $x = x''$; $k = 1$
8:     else
9:       $k = k + 1$
10:    end if
11:  end while
12: end while
```

Fig. 1. Basic VNS

In order to reduce the computational complexity we use the filtration of edges presented in [13]. The idea of this method is the following. If the lower bound of the objectives of 1 on all communication trees, which contain the edge e, exceeds the objective on another known feasible solution then the edge e is removed from the communication graph. This filtration is applied to the communication graph as soon as new record solution has been obtained. For the first approximation of our heuristic we generate two trees: one by MST and another by IPP, and then we take the better of them.

```
1: Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, ..., k_{\max}$ that will be used
   for the shaking phase, and let $\mathcal{N}_0 = \{x\}$; select the set of neighborhood structures
   $N_l$, for $l = 1, ..., l_{\max}$ that will be used for local search; find an initial solution $x$;
   set $k = 0$;
2: while the stopping criteria is not met do
3:    while $k \leq k_{\max}$ do
4:       Perform Shaking: generate a point $x'$ at random from $\mathcal{N}_k(x)$;
5:       while $l \leq l_{\max}$ do
6:          Find the best solution $x'' \in N_l(x')$.
7:          if $x''$ is better than $x'$ then
8:             $x' = x''$; $l = 1$
9:          else
10:            $l = l + 1$
11:         end if
12:      end while
13:      if $x'$ is better than $x$ then
14:         $x = x'$; $k = 1$
15:      else
16:         $k = k + 1$
17:      end if
18:   end while
19: end while
```

Fig. 2. General VNS

Local Searches. Each neighborhood structure of the local search phase of VNS-based heuristics is used only for local search. Therefore, the descriptions of the local search procedures are sufficient for the definition of the corresponding neighborhood structures, and there is no necessity for explicit formulation of the neighborhood structures.

We propose two local search heuristics which perform edge switchings, but, as opposed to the known ES-like heuristics, they do not perform each edge switching in the best way, but instead of this they iteratively consider a list of edges and perform the best switching for each considered edge. The procedure stops if at some iteration there was no any improvements in all steps of the loop over the edges. There are two possible variants of this approach, we called them Adding and Best Removing (ABR) and Removing and Best Adding (RBA). The pseudo-codes of these local searches can be found, respectively, in Fig. 3 and in Fig. 4. Note that these local searches are similar to ES1a and ES1b from [14], but, as opposed to ES1a and ES1b, ABR and RBA guarantee that the obtained solution is a local optimum.

Shaking. For the shaking procedure, which is used in basic VNS and general VNS, we propose two algorithms. The first one is random shaking which is described in Fig. 5. It consists of sequence of random edge addings and random edge removings. The second algorithm is intensified shaking (Fig. 6), which adds a random edge at first and then removes an edge from the cycle whose deletion

```
1: Input: $G = (V, E)$ - communication graph, $T = (V, F)$ — spanning tree;
2: improved = true;
3: while improved do
4:    improved = false;
5:    $G = FilterEdges(G, W(T))$;
6:    $D = E \setminus F$;
7:    for each edge $e \in D$ do
8:       Find the such edge $f$ in a cycle of $F \cup \{e\}$, whose removing leads to the
         maximum decrease of the objective;
9:       $T' = (V, F \cup \{e\} \setminus \{f\})$;
10:      if $W(T') < W(T)$ then
11:         $T = T'$;
12:         improved = true;
13:      end if
14:   end for
15: end while
```

Fig. 3. ABR local search

```
1: Input: $G = (V, E)$ - communication graph, $T = (V, F)$ — spanning tree;
2: $G = FilterEdges(G, W(T))$;
3: improved = true;
4: while improved do
5:    improved = false;
6:    for each edge $e \in F$ do
7:       Let $A$ and $B$ be the edges of connected components obtained after removing
         of $e$ from $T$;
8:       Find such edge $f \in E$ which connects $A$ and $B$ and whose adding to $A \cup B$
         leads to the minimum increase of the objective;
9:       $T' = (V, A \cup B \cup \{f\})$;
10:      if $W(T') < W(T)$ then
11:         $T = T'$;
12:         $G = FilterEdges(G, W(T))$;
13:         improved = true;
14:      end if
15:   end for
16: end while
```

Fig. 4. RBA local search

reduces the objective at most. In both algorithms replacing of edges is repeated k times. Let $k_{\max}$ be the maximum number of edge replacements by the shaking procedure. Note that k_{max} is a free parameter in the considered VNS heuristics, and its best value of this parameter is estimated experimentally.

```
1: Input: $G = (V, E)$ - communication graph, $T = (V, F)$ – spanning tree, $k$ – neighborhood index;
2: $i = 1$;
3: while $i \leq k$ do
4:     Select an edge $e_1 \in E \setminus F$ at random;
5:     $F' = F \cup \{e_1\}$;
6:     Let $C \subseteq F'$ be a cycle containing $e_1$; select at random an edge $e_2 \in C$;
7:     $F' = F' \setminus \{e_2\}$;
8:     $T = (V, F')$;
9: end while
```

Fig. 5. Random shaking

```
1: Input: $G = (V, E)$ - communication graph, $T = (V, F)$ — spanning tree, $k$ — neighborhood index;
2: $i = 1$;
3: while $i \leq k$ do
4:     Select an edge $e_1 \in E \setminus F$ at random;
5:     $F' = F \cup \{e_1\}$;
6:     Let $C \subseteq F'$ be a cycle containing $e_1$; select an edge $e_2 \in C$ whose deletion reduces the objective at most;
7:     $F' = F' \setminus \{e_2\}$;
8:     $T = (V, F')$;
9: end while
```

Fig. 6. Intensified shaking

5 Simulation

All the proposed algorithms have been implemented in C++ using the Visual Studio 2010 Integrated Development Environment. A simulation was executed for $n = 10, 30, 50, 250$, and in some cases for $n = 500$. For the same dimension, 100 different instances were randomly generated. For each instance, a required number of points was uniformly scattered on a square area with a side of 10 units. After this, a complete edge-weighted graph whose vertices correspond to the points and whose edge weights were equal to the squared distances between the points was defined. Then the calculation of MST and IPP were run on the complete graph, and the best of the obtained two trees was chosen as the first approximation solution for the heuristics. The experiment was performed on an Intel Core i5-4460 (3.2 GHz) 8 Gb machine, and only one thread was used at the same time for all algorithms except CPLEX and GA.

In order to compare the algorithms for the large dimensions, when an optimal solution cannot be found in acceptable time, we calculated the average improvement compared to MST. This estimate was often used for these purposes in the related papers [1,14,17]. For the small dimensions ($n \leq 30$), we defined the parameters of the problem formulation as an integer linear programming problem (ILP), as proposed in [3], and then we obtained the optimal solution using the IBM ILOG CPLEX package. In Table 1 the improvement of optimal solution over MST and the average CPLEX CPU time are presented. Currently neither of known packages and ILP formulations allow to obtain an optimal solution for $n \geq 40$ in acceptable time [1,3,13]. Note that we have parallelized CPLEX on 4 threads to speed-up calculations.

Table 1. CPLEX (optimal solution). Improvement over MST and CPU time

n	Impr. to MST	CPU time
10	3.98 %	0.33 s
30	5.78 %	93.53 s

For the VNS-based heuristics, it is necessary to define the parameter $k_{\max}$. For this goal, each algorithm was run on the same instances with different values of $k_{\max}$. It appeared that, beginning from $k_{\max} = 30$, on average, the objective of the obtained solution did not decrease significantly, whereas the runtime grown fast. Moreover, on average, the runtime of all the algorithms remained accessible for $k_{\max} = 30$. Therefore, in all the VNS-based algorithms, we set $k_{\max} = 30$.

In the Table 2 the effect of filtration from [13] is presented. The first column represents the percentage of edges removed after applying the filtration procedure to the complete graph when the results of MST and IPP are known. In the other columns the speed-ups of some of the heuristics are reflected. One can see that filtration significantly simplifies the initial graph and speeds up the algorithms. In all further results all heuristics use filtration procedure.

Table 2. Filtration effect

n	Filtered edges	Speed-up of ES	Speed-up of ABR	Speed-up of RBA
30	53.02 %	53.08 %	58.96 %	46.15 %
50	55.81 %	59.62 %	63.36 %	53.74 %
100	59.71 %	66.38 %	64.3 %	59.57 %
250	60.84 %	70.59 %	66.53 %	62.73 %

In Table 3 the local search algorithms are compared. The best values are marked bold. The best solutions were always obtained by EFS, but its running time increases very fast with growing n, and it works more than 1000 s already on 200 nodes.

Table 4 represents CPU time and improvement over MST of the basic VNS with different local search procedures and random shaking. In Table 5 the results obtained by the same algorithms but with intensified shaking are presented. Since, on average, intensified shaking works slightly better that the random shaking, in the further tables intensified shaking is used in VNS-based heuristics.

Table 3. Local search heuristics. Improvement over MST and CPU time.

n	ABR		RBA		EFS		ES		LI	
	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time
10	3.76 %	0.00 s	3.72	0.00 s	**3.96** %	0.00 s	3.75 %	0.00 s	3.04 %	0.00 s
30	5.03 %	0.00 s	5.05 %	0.00 s	**5.58** %	0.09 s	5.07 %	0.00 s	3.56 %	0.00 s
50	5.35 %	0.00 s	5.33 %	0.00 s	**6.08** %	0.92 s	5.45 %	0.00 s	3.98 %	0.00 s
100	5.52 %	0.02 s	5.5 %	0.02 s	**6.17** %	29.39 s	5.59 %	0.05 s	3.77 %	0.00 s
250	5.61 %	0.26 s	5.6 %	0.23 s	–	–	5.71 %	1.09 s	3.94 %	0.01 s

Table 4. Basic VNS with random shaking. Improvement over MST and CPU time.

n	B_ABR		B_RBA		B_ES		B_LI	
	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time
10	**3.98** %	0.00 s	3.96 %	0.00	**3.98** %	0.01 s	3.93 %	0.00 s
30	5.74 %	0.06 s	**5.78** %	0.06	5.76 %	0.08 s	4.40 %	0.00 s
50	6.21 %	0.24 s	6.29 %	0.26	**6.30** %	0.32 s	3.96 %	0.00 s
100	6.08 %	1.39 s	**6.23** %	1.71	6.11 %	1.81 s	3.50 %	0.01 s
250	6.12 %	17.97 s	**6.27** %	22.52	6.00 %	19.27 s	3.62 %	0.02 s

Table 5. Basic VNS with intensified shaking. Improvement over MST and CPU time.

n	B_ABR		B_RBA		B_ES		B_LI	
	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time
10	3.94 %	0.00 s	**3.98** %	0.00 s	**3.98** %	0.00 s	3.77 %	0.00 s
30	5.71 %	0.05 s	**5.77** %	0.05 s	5.76 %	0.06 s	4.14 %	0.00 s
50	6.16 %	0.23 s	6.26 %	0.21 s	**6.27** %	0.21 s	3.82 %	0.00 s
100	6.02 %	1.44 s	**6.24** %	1.44 s	6.12 %	1.21 s	3.42 %	0.01 s
250	6.01 %	15.41 s	**6.27** %	21.46 s	5.96 %	12.09 s	3.45 %	0.02 s

The general VNS-based heuristics results presented in Table 6. We have run two variants of general VNS. Both of them used ABR and RBA as local searches and intensified shaking. G_AR is general VNS where in each iteration of the local search phase ABR was run at first and RBA was run next. G_RA is general VNS in each iteration of the local search phase RBA was run at first and ABR was run next.

In Table 7 the results obtained by ILS with different local searches are presented. The random increase mutation was used in ILS and, as well as it was done in [17], 200 iterations were run before stop.

Table 6. General VNS with intensified shaking. Improvement over MST and CPU time.

n	G_AR		G_RA	
	Impr. to MST	CPU time	Impr. to MST	CPU time
10	**3.98** %	0.00 s	3.96 %	0.00 s
30	5.74 %	0.07 s	**5.78** %	0.08 s
50	6.15 %	0.28 s	**6.29** %	0.29 s
100	6.03 %	1.74 s	**6.20** %	1.86 s
250	6.05 %	21.27 s	**6.30** %	31.14 s

Table 7. ILS-based heuristics. Improvement over MST and CPU time.

n	ILS_ABR		ILS_RBA		ILS_ES		ILS_LI	
	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time
10	**3.98** %	0.02 s	3.9 %	0.02 s	**3.98** %	0.02 s	3.17 %	0.01 s
30	5.72 %	0.32 s	5.75 %	0.33 s	**5.78** %	0.43 s	2.929 %	0.02 s
50	6.23 %	1.111 s	6.28 %	1.23 s	**6.33** %	1.73 s	3.187 %	0.04 s
100	6.12 %	6.353 s	6.23 %	7.96 s	**6.31** %	13.33 s	3.048 %	0.09 s
250	6.06 %	65 s	6.16 %	107.5 s	**6.4** %	250 s	3.21 %	0.29 s

Although, on average, ILS outperforms the VNS-based heuristics, it requires significantly more time, and the average excesses of the best of ILS-based heuristic ILS_ES over the best of VNS-based metaheuristics B_RBA and G_RA are not so significant — they never exceed 0.1 %. Therefore, we compared the solution obtained by one of the best VNS-based metaheuristics, basic VNS with RBA and intensified shaking (B_RBA), with the solution obtained by ILS_ES in the same running time as B_RBA. These results are presented in Table 8. In the same manner, we have compared the G_RA (which appeared to be the best of general VNS-based heuristics) with ILS_ES, see Table 9. Except the improvement over MST, for each of two heuristics B_RBA and G_RA, we calculated the percentage of cases when its solution is better than ILS and the percentage of cases when it is worse than ILS_ES. One can see that, on average, B_RBA and G_RA both outperform ILS_ES, especially on large dimensions. The advantages of the both VNS-based heuristics are most strongly shown when $n = 500$. In this case B_RBA yielded more accurate solution than ILS_ES in 99 % of cases, the average improvement of B_RBA over MST exceeds the same estimation of LI_ES by 0.44 % which is about 7.5 % of the improvement, and the maximum improvement over to MST exceeds the same estimation of LI_ES by 0.82 % which is 10.16 % of the improvement. The results obtained by G_RA in the case of $n = 500$ are very impressive as well: G_RA yields better solution than ILS_ES in 94 % of cases, its average excess of improvement over MST is 0.38 %, which is 6.37 %

Table 8. Comparison of the results for the best of the basic VNS-based heuristics B_RBA and for the best of the iterated local search-based algorithms ILS_ES.

n	B_RBA is better	ILS_ES is better	B_RBA: Impr. to MST			ILS_ES: Impr. to MST			CPU time
			Min	Avg	Max	Min	Avg	Max	
10	2 %	0 %	0 %	**3.98** %	19.82 %	0 %	3.95 %	19.82 %	0.00 s
30	11 %	2 %	0.82 %	**5.78** %	15 %	0.82 %	5.7 %	14.58 %	0.05 s
50	30 %	10 %	1.20 %	**6.28** %	13.56 %	1.20 %	6.2 %	13.41 %	0.20 s
100	54 %	26 %	2.48 %	**6.23** %	10.42 %	2.38 %	6.15 %	10.86 %	1.43 s
250	85 %	15 %	3.64 %	**6.29** %	9.50 %	3.46 %	6.04 %	9.36 %	22.63 s
500	99 %	1 %	4.10 %	**6.34** %	8.89 %	3.72 %	5.90 %	8.07 %	208.2 s

Table 9. Comparison of the results for the best of the general VNS-based heuristics G_RA and for the best of the iterated local search-based algorithms ILS_ES.

n	G_RA is better	ILS_ES is better	G_RA: Impr. to MST			ILS_ES: Impr. to MST			CPU time
			Min	Avg	Max	Min	Avg	Max	
10	0 %	0 %	0 %	**3.98** %	19.82 %	0 %	**3.98** %	19.82 %	0.01 s
30	6 %	6 %	0.82 %	**5.77** %	15 %	0.82 %	5.74 %	14.58 %	0.08 s
50	17 %	14 %	1.2 %	**6.29** %	13.66 %	1.2 %	**6.29** %	13.56 %	0.30 s
100	47 %	36 %	1.87 %	6.21 %	10.34 %	2.42 %	**6.23** %	10.86 %	2.11 s
250	72 %	28 %	3.69 %	**6.29** %	9.49 %	3.65 %	6.16 %	9.60 %	31.42 s
500	94 %	6 %	4.30 %	**6.35** %	8.6 %	3.75 %	5.97 %	8.29 %	279.7 s

of the improvement. It should be noted, that LI_ES had appeared to be too time-consuming in a case of $n = 500$. Its average running time on 10 instances exceeded 1200 s.

In [2] two hybrid genetic algorithms for the MPSCP were proposed. The best results had been obtained by the genetic algorithm which used VND-based heuristic as mutation. In Table 10 the results of this hybrid genetic algorithm GA_VND are compared with the best VNS-based heuristics: B_RBA and G_RA. One can see that GA_VND solved the problem significantly faster, but it should be taken into account that it was well parallelized and used four parallel threads. However, VNS-based heuristics yield more accurate solutions, especially on large

Table 10. Comparison of the results for the best of VNS-based heuristics and hybrid genetic algorithm GA_VND.

n	B_RBA		G_RA		GA_VND	
	Impr. to MST	CPU time	Impr. to MST	CPU time	Impr. to MST	CPU time
10	**3.98** %	0.00 s	**3.98** %	0.01 s	**3.98** %	0.06 s
30	**5.78** %	0.05 s	5.77 %	0.08 s	5.75 %	0.14 s
50	6.28 %	0.20 s	**6.29** %	0.3 s	6.20 %	0.31 s
100	**6.23** %	1.427 s	6.21 %	2.11 s	5.96 %	1.12 s
250	**6.29** %	22.63 s	**6.29** %	31.42 s	5.87 %	6.35 s
500	6.34 %	208.2 s	**6.35** %	279.7 s	5.71 %	31.8 s

instances: in a case of $n = 500$ their average improvement over MST exceeds the same estimation for the GA_VND by more than 0.6 %, which is 10.5 % of the improvement. Since GA_VND was stopped after stabilization (when the quality of solutions was not changed during the last 20 iterations), we did not expect that its solutions would become significantly better after the longer work. Therefore, GA_VND was not compared with the proposed VNS-based heuristics by time limit, as it was done for ILS_ES.

6 Conclusion

In this paper we have presented new variable neighborhood search-based heuristics for the Minimum Power Symmetric Connectivity Problem. We used two known variants of the VNS metaheuristic: basic VNS and general VNS. As local search we used already known heuristics ES, EFS and LI as well as two new heuristics: ABS and ARB. We also used filtration of edges of the communication graph inside our algorithms in order to reduce the computation time. The numerical experiment has shown that the best of the proposed VNS-based heuristics (namely, B_RBA and G_RA) are more suitable to use in practice than the best of known algorithms (iterated local search-based algorithm proposed in [17] and hybrid genetic algorithm proposed in [2]): on average, our heuristics obtain significantly more accurate solutions in short time and allow to successfully get solutions very close to optimal in large dimension cases. In future we plan to implement the variable neighborhood decomposition search [11] for this problem in order to solve it in larger dimensions in acceptable time.

Acknowledgments. The research of A. Erzin is partly supported by the Russian Foundation for Basic Research (grant no. 16-07-00552). The research of N. Mladenovic is partly supported by the Ministry of Education and Science, Republic of Kazakhstan (Institute of Information and Computer Technologies) (project no. 0115PK00546). The research of R. Plotnikov is supported by the Russian Foundation for Basic Research (grant no. 16-37-60006).

References

1. Althaus, E., Calinescu, G., Mandoiu, I.I., Prasad, S.K., Tchervenski, N., Zelikovsky, A.: Power efficient range assignment for symmetric connectivity in static ad hoc wireless networks. Wireless Netw. **12**(3), 287–299 (2006)
2. Erzin, A., Plotnikov, R.: Using VNS for the optimal synthesis of the communication tree in wireless sensor networks. Electron. Notes Discrete Math. **47**, 21–28 (2015)
3. Erzin, A., Plotnikov, R., Shamardin, Y.: On some polynomially solvable cases and approximate algorithms in the optimal communication tree construction problem. J. Appl. Indust. Math. **7**, 142–152 (2013)
4. Fuchs, B.: On the hardness of range assignment problems. Technical report. TR05-113, Electronic Colloquium on Computational Complexity (2005)

5. Clementi, A.E.F., Penna, P., Silvestri, R.: Hardness results for the power range assignment problem in packet radio networks. In: Hochbaum, D.S., Jansen, K., Rolim, J.D.P., Sinclair, A. (eds.) RANDOM 1999 and APPROX 1999. LNCS, vol. 1671, pp. 197–208. Springer, Heidelberg (1999)
6. Calinescu, G., Mandoiu, I.I., Zelikovsky, A.: Symmetric connectivity with minimum power consumption in radio networks. In: Baeza-Yates, R.A., Montanari, U., Santoro, N. (eds.) Proceedings of the 2nd IFIP International Conference on Theoretical Computer Science. IFIP Conference Proceedings, vol. 223, pp. 119–130. Kluwer, Dordrecht (2002)
7. Cheng, X., Narahari, B., Simha, R., Cheng, M.X., Liu, D.: Strong minimum energy topology in wireless sensor networks: NP-completeness and heuristics. IEEE Trans. Mob. Comput. **2**(3), 248–256 (2003)
8. Chu, T., Nikolaidis, I.: Energy efficient broadcast in mobile ad hoc networks. In: Proceedings of AD-HOC Networks and Wireless (2002)
9. Hanafi, S., Lazic, J., Mladenovic, N., Wilbaut, C., Crevits, I.: New variable neighbourhood search based 0–1 MIP heuristics. Yugoslav J. Oper. Res. (2015). doi:10.2298/YJOR140219014H
10. Hansen, P., Mladenovic, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**, 449–467 (2001)
11. Hansen, P., Mladenovic, N., Perez-Britos, D.: Variable neighborhood decomposition search. J. Heuristics **7**(4), 335–350 (2001)
12. Kirousis, L.M., Kranakis, E., Krizanc, D., Pelc, A.: Power consumption in packet radio networks. Theoret. Comput. Sci. **243**(1–2), 289–305 (2000)
13. Montemanni, R., Gambardella, L.M.: Exact algorithms for the minimum power symmetric connectivity problem in wireless networks. Comput. Oper. Res. **32**(11), 2891–2904 (2005)
14. Park, J., Sahni, S.: Power assignment for symmetric communication in wireless networks. In: Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC), Washington, pp. 591–596. IEEE Computer Society, Los Alamitos (2006)
15. Rappaport, T.S.: Wireless Communications: Principles and Practices. Prentice Hall, Upper Saddle River (1996)
16. Santi, P.: Topology Control in Wireless Ad Hoc and Sensor Networks. Wiley, Chichester (2005)
17. Wolf, S., Merz, P.: Iterated local search for minimum power symmetric connectivity in wireless networks. In: Cotta, C., Cowling, P. (eds.) EvoCOP 2009. LNCS, vol. 5482, pp. 192–203. Springer, Heidelberg (2009)

Author Queries

Chapter 18

Query Refs.	**Details Required**	**Author's response**
AQ1	Please confirm if the corresponding author and mail id is correctly identified. Amend if necessary.	

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

Instruction to printer	*Textual mark*	*Marginal mark*
Leave unchanged	• • • under matter to remain	Ⓥ
Insert in text the matter indicated in the margin	⋏	New matter followed by ⋏ or ⋏ⓧ2
Delete	/ through single character, rule or underline or ⊢⊣ through all characters to be deleted	∂/ or ∂/ⓧ2
Substitute character or substitute part of one or more word(s)	/ through letter or ⊢⊣ through characters	new character / or new characters /
Change to italics	— under matter to be changed	⌞
Change to capitals	≡ under matter to be changed	≡
Change to small capitals	= under matter to be changed	=
Change to bold type	~ under matter to be changed	~
Change to bold italic	≂ under matter to be changed	≂
Change to lower case	Encircle matter to be changed	≢
Change italic to upright type	(As above)	⊥
Change bold to non-bold type	(As above)	⊥
Insert 'superior' character	/ through character or ⋏ where required	Y or ⋏ under character e.g. $\overset{2}{Y}$ or $\overset{2}{⋏}$
Insert 'inferior' character	(As above)	⋏ over character e.g. $\underset{2}{⋏}$
Insert full stop	(As above)	⊙
Insert comma	(As above)	,
Insert single quotation marks	(As above)	$\overset{‘}{Y}$ or $\overset{‘}{⋏}$ and/or $\overset{’}{Y}$ or $\overset{’}{⋏}$
Insert double quotation marks	(As above)	$\overset{“}{Y}$ or $\overset{“}{⋏}$ and/or $\overset{”}{Y}$ or $\overset{”}{⋏}$
Insert hyphen	(As above)	⊢⊣
Start new paragraph	⌐	⌐
No new paragraph	↩	↩
Transpose	⊔⊓	⊔⊓
Close up	linking ⁐ characters	⁐
Insert or substitute space between characters or words	/ through character or ⋏ where required	Y
Reduce space between characters or words	| between characters or words affected	⊤