Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in DISCRETE MATHEMATICS

Electronic Notes in Discrete Mathematics 47 (2015) 21–28

www.elsevier.com/locate/endm

Using VNS for the Optimal Synthesis of the Communication Tree in Wireless Sensor Networks

A.I. Erzin [a,1] R.V. Plotnikov [b,2]

[a] *Sobolev Institute of Mathematics and Novosibirsk State University, Novosibirsk, Russia*

[b] *Novosibirsk State University, Novosibirsk, Russia*

Abstract

We investigate the NP-hard problem of finding an optimal spanning tree in a given undirected weighted graph, which occurs while minimising the power consumption of data transmission in radio networks. We proposed new heuristics and conducted an a posteriori analysis. All of the proposed methods showed high effectiveness, but it is worth noting a hybrid genetic algorithm using variable neighbourhood search (VNS) as mutations.

Keywords: Wireless Sensor Networks, Adjustable Communication Range, Spanning Tree, Heuristics.

1 Introduction and Formulation of the Problem

Elements of many systems use wireless communication. Thus, the energy consumption of the element is proportional to the transmission distance [1]. In

[1] Email: adilerzin@math.nsc.ru

[2] Email: nomad87@ngs.ru

http://dx.doi.org/10.1016/j.endm.2014.11.004

wireless sensor networks (WSNs), the sensors have a limited supply of energy and energy efficiency for extending the network's lifetime [10]. Modern sensors are able to adjust the transmission range, but there is a problem with defining the range of each sensor to minimize the total energy consumption and to maintain a connected graph. As a rule, a graph from which the communications network is sought is considered to be complete [1,2,4,9]. However, the signal does not always propagate equally in all directions. Therefore, in general, one should assume that the communication graph is arbitrary and that the energy consumption to maintain the existence of the edges do not only depend on the distance between the corresponding nodes [5].

The problem of determining the transmission distance of each vertex that is located in a Euclidean space to induce a strongly connected graph in which the overall energy to communicate is minimal was investigated in [9]. The authors of [1] proposed an algorithm with an asymptotic ratio of 5/3, a polynomial algorithm constructing an 11/6-approximate solution as well as the exact algorithm – branch and bound method, which uses a new formulation of the problem as a linear integer programming problem (LIP). In the paper [2], the problem of determining the transmitter capacities for transmitting data between the two distances, "small" and "long", was considered. The NP-hardness of this problem is proved. In [5], a more accurate ratio for the minimum spanning tree is proposed and the polynomially solvable special cases of the problem are found.

We consider the

Problem. *The simple undirected weighted graph $G = (V, E)$ with a vertex set V, $|V| = n$, and a set of edges E is given. Let $c_{ij} \geq 0$ be the* weight *of the edge $(i, j) \in E$. We want to find a spanning tree T^* of the graph G, which is the solution of the 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 a vertex i in the tree T.

Problem (1) is known as the *min-power symmetric connectivity problem* [1]. Further, any feasible solution of (1) – spanning tree – will also be called a *communication* tree. The considered problem is strongly NP-hard [1,5,6,9]. Moreover, if $N \neq NP$, then there is no polynomial algorithm yielding a $\left(1 + \frac{1}{260}\right)$-approximate solution [5,6]. It is known that the minimum spanning tree P is a 2-approximate solution of the problem (1). Moreover, [5] proved that $W(P)/W(T^*) \leq 2 - 2a/(a + b + 2b/(n - 2))$, where the weights of the edges of P are in the interval $[a, b]$.

In this paper, we propose new heuristic algorithms and present the numerical experiments that show their efficiency. In particular, we perform a comparison of the algorithms with and without VNS [7,8].

2 Heuristic Algorithms

The first algorithm (LI) is the procedure of local improvements for the arbitrary spanning tree. The second algorithm is a local search with variable neighbourhood searches (VNS). The third algorithm is a hybrid genetic algorithm GA_LI, which uses LI as a mutation. The fourth algorithm is a hybrid genetic algorithm GA_VNS that uses VNS as a mutation. Let us first describe the procedures LI and VNS and then propose the general scheme of the genetic algorithm, which is used in the algorithms GA_LI and GA_VNS.

2.1 Local Improvements

Initial Step. Let T be the arbitrary spanning tree. Denote $d_i = \max\limits_{j \in V_i(T)} c_{ij}$ as the weight of the vertex $i \in V$. Choose the vertex $v_0 = \arg\min\limits_{i \in V} d_i$ as a root of the tree T. Define the orientation of the edges of the tree from the root. Denote $p(i)$ as the parent vertex for the node $i \in V$ (i.e., the arc $(p(i), i) \in T$) and suppose that $p(v_0) = v_0$. We call $D_{(i,j)} = 2c_{ij} - \min\{c_{ij}, \max\limits_{k \in N_i(T) \setminus \{j\}} c_{ik}\} - \min\{c_{ij}, \max\limits_{k \in N_j(T) \setminus \{i\}} c_{jk}\}$ the *deterioration* for the edge (i, j). Calculate the deteriorations for all edges of the tree T.

Basic Step. Look through all the vertices except the root of the tree in increasing order of the deteriorations $D_{(p(i),i)}$ and for each vertex $i \in V \setminus \{v_0\}$, execute the following local improvement procedure. Delete the edge $(p(i), i)$ and break tree T into two connected components. For each vertex j of the component containing the root v_0 (denote this component as $S(i)$), perform the following procedure. Add the edge (j, i) to the graph $T \setminus \{(p(i), i)\}$ and calculate the deterioration $D_{(j,i)}$. Then, delete the edge (j, i) from the tree and repeat the procedure for another node in $S(i)$. Connect the components using the edge with the smallest deterioration. If no edge has less deterioration than the edge $(p(i), i)$, then $(p(i), i)$ remains in the tree T. Note that when replacing the edge, the objective function (1) decreases.

If after reviewing all vertices with at least one change for the edge, repeat the basic step. Otherwise, exit the procedure. The constructed tree is the desired approximate solution of the problem (1).

2.2 Variable Neighbourhood Search (VNS)

The idea using different rules for constructing neighbourhoods for the current solution was proposed by P. Hansen and N. Mladenović [7,8]. Let tree T be the solution to the problem (1) (i.e., some spanning tree). We consider the following system of neighbourhoods $\{N_k(T)\}_{k\in\mathbb{Z}^+}$ for T. Each set $N_k(T)$ consists of the spanning trees, and the difference from T is not more than k edges. An iteration of the algorithm consists of a successive local search in the neighbourhoods N_1, N_2 and N_3. For each $k \in \{1,2,3\}$, the local improvement procedure is as follows. First, with a probability proportional to the contribution of the objective function (contribution of the edge (i,j) is decreasing for $W(T)$ after the deletion of (i,j)), remove k edges, while the tree T splits into $k+1$ connected component. Then, search for the edges connecting the $k+1$ components in a spanning tree (without changing the edges inside the component) to obtain the best tree. If $k \leq 2$, then we perform an exhaustive search of the solutions in the neighbourhoods. If $k = 3$, then the enumeration of possibilities is too complex. Therefore, a new tree is constructed as follows. First, for every pair of the components sought, an edge connects them with a minimum contribution to the objective function value (denote the set of these edges as U). Then, the min-weight tree is built for the complete graph with vertices corresponding to the connected components, and the edge set is U. If some $k \in \{1,2,3\}$ procedure fails to find a better tree, then proceed to the next neighbourhood $N_{next(k)}(T)$, where $next(k) = k+1$ for $k \in \{1,2\}$) and $next(3) = 1$. If successive attempts to improve the solution in the neighbourhoods $N_1(T)$, $N_2(T)$ and $N_3(T)$ have been unsuccessful, the algorithm stops.

2.3 Hybrid Genetic Algorithm

For an approximate solution to problem (1), we propose a genetic algorithm, which uses the procedures described above as mutations. The algorithm takes a succession of populations, each of which is a set of spanning trees of a graph G. The algorithm terminates after the stopping criterion, which is described below. The input algorithm has the following parameters: $N \in \mathbb{Z}^+$ – population size; $M \in \mathbb{Z}^+$ – the number of new individuals in the population after each iteration; $P \in [0,1]$ – the probability of mutation; $K \in \mathbb{Z}^+$ – the stopping criterion parameter.

Initial Step. The initial population is generated by taking the minimum spanning tree, and the remaining $N-1$ individuals are constructed as follows. First, the root node is randomly selected, and then one random edge is

sequentially added and joins a vertex of the tree-to-be with the new vertex. The probability of selecting the edge is inversely proportional to its weight. The obtained individual is added to a population in the case if it is not a copy of an existing individuals.

Iteration.

(i) Calculate the fitness of each individual T as $1/W(T)$.

(ii) Sequentially select pairs for crossing from the population M. The probability of selecting individual is proportional to the individual's fitness, and each individual can belong to several couples. A cross between two individuals is one descendant.

(iii) When crossing two individuals, one arbitrary vertex is selected as a root in both trees, and the edges obtain the orientation from the root. The set of edges includes all descendant edges that appear in both trees. To add other edges, select a vertex that is not at the end of the arc in the tree-to-be. Each of the two parents has exactly one edge ending at the selected vertex. One of these two edges is randomly selected with a probability inversely proportional to the contribution of the edge in the objective function.

(iv) At the stage of mutation, the procedure of local improvements is applied to each child with probability P. The resulting individual is added to the population if it is not a copy of existing individuals.

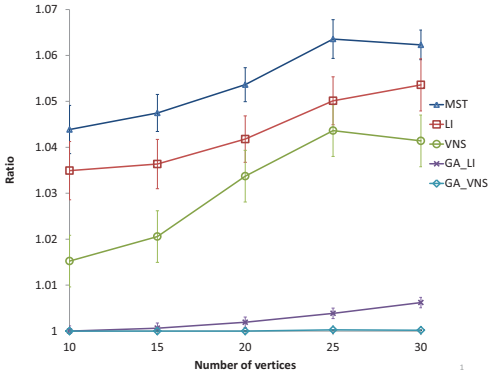
(v) N fittest individuals are selected in a new population.

Stopping Criterion. The algorithm stops if the last K iterations do not change the minimum and maximum values for the objective function of the elements of the population.

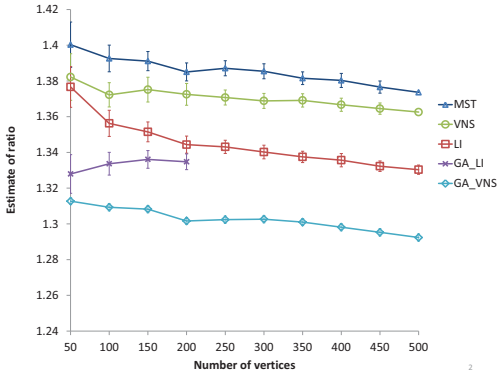
3 Numerical Experiment

We used the LIP formulation of (1) from [5] to find the *optimal* solution using the CPLEX package. The experiment was conducted for $n \in [5, 500] \bigcap \mathbb{Z}$. For the same dimension, 100 different instances were generated. A hybrid genetic algorithm was run with different values of the input parameters. As a result, the combination of parameters $N = 30, M = 20, P = 0.8$, and $K = 20$ on average yield the most near-optimal solution in a reasonable time. At these values, the parameters of the genetic algorithm results were compared with the trees built by other methods. The experiment was conducted on a computer with an Intel Core i5-3470 (3.2GHz) with 8Gb. The results are shown in the

figures below.

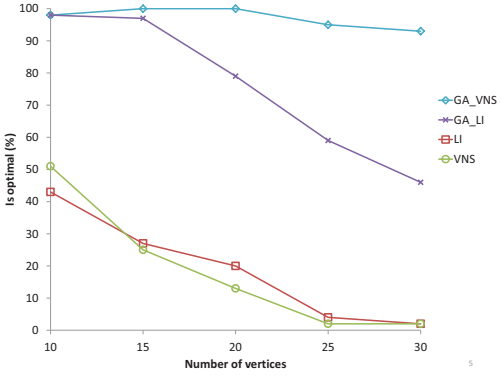


a) Ratio

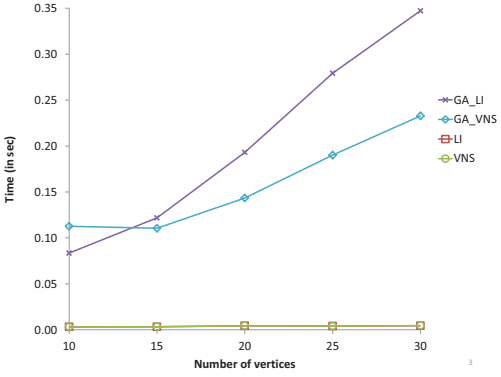


b) Upper bound ratio

Fig. 1.



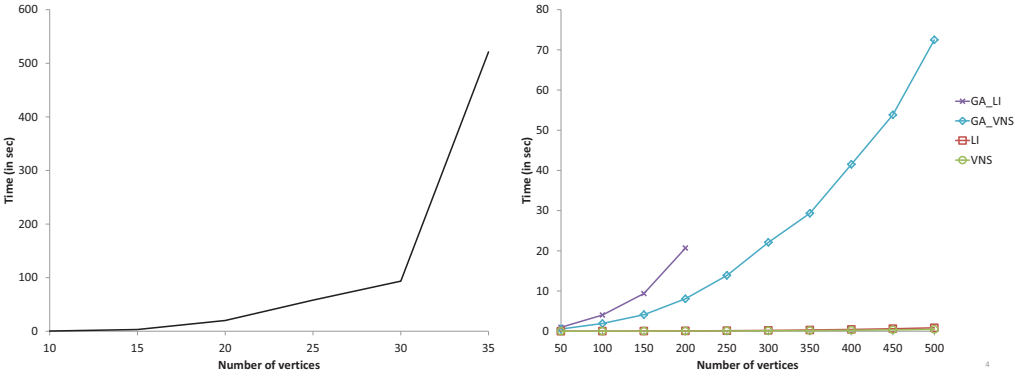
a) Percentage of cases where the algorithm constructs an optimal solution



b) The running time of heuristics

Fig. 2.

Fig. 1 compares the ratios $W_A(T)/W(T^*)$, where $W_A(T)$ is the value of the objective function for the solution yielded by algorithm A. On average, in those cases when it was possible to accurately calculate the ratio (i.e., when $n \leq 35$), the algorithm GA_LI built a solution which differs from the optimum by no more than 0.6% and the algorithm GA_VNS built a solution that differed from the optimal value by not more than 0.01%. The LI algorithm proved to be more accurate than the VNS algorithm when $n > 50$. When $n > 50$, CPLEX failed to find an optimal solution within a reasonable time, and instead of $W(T^*)$ we used the lower bound value of the optimal objective function. In Fig. 2(a), we see that at a small dimension, the algorithm



a) The running time of CPLEX b) The running time of heuristics

Fig. 3.

GA_VNS almost always built the optimal solution, and in the worst case (for $n = 35$), the algorithm GA_VNS built an optimal solution in 98% of cases, while the algorithm GA_LI only built an optimal solution in 43% of cases.

The graphs in Fig. 2(b) and Fig. 3 represent the running time of the algorithms. The algorithm GA_LI solves the problem in a reasonable time when $n \leq 200$, the algorithm GA_VNS – when $n \leq 500$, and the CPLEX package – when $n \leq 35$. Algorithms LI and VNS yield an approximate solution for $n = 500$ in less than 1 second.

4 Conclusion

The numerical experiment showed high efficiency for the proposed algorithms. An approximate solution yielded by the hybrid genetic algorithm using VNS as mutations was the closest to the optimal solution. The experiment showed that algorithm GA_VNS solves the problem in a reasonable amount of time when $n \leq 500$ and has a lower growth rate for the operating time than algorithm GA_LI. LI is applied to the minimum spanning tree and has high efficiency. In large dimensions, the application of this method is most justified for achieving high-speed calculations and for ensuring the quality of the solution. The use of the package CPLEX for the LIP also proved to be an effective way to solve the problem for a small dimension. Thus, the average problem was solved by CPLEX in less than 100 seconds for $n = 30$ (with parallelization on 4 threads) and in 520 seconds for $n = 35$.

The most effective algorithm of the proposed algorithms was the GA_VNS (hybrid genetic algorithm using VNS as mutations) yielding a 1.0009-approximate

solution for small dimensions ($n \leq 35$) and a 1.3-approximate solution for $50 \leq n \leq 500$. This gap occurs because, for the assessment of the ratio for a large dimension, the lower bound was used instead of the optimal value for the objective function. In fact, we assume that the 1.0009-approximate solution is built for each dimension of the problem.

References

[1] Althaus, E., G. Calinescu, I. Mandoiu, S. Prasad, N. Tchervenski and A. Zelikovsky, *Power Efficient Range Assignment for Symmetric Connectivity in Static Ad Hoc Wireless Networks*, Wireless Networks **12** (2006), pp. 287–299.

[2] Carmi P. and M. L. Katz, *Power Assignment in Radio Networks with Two Power Levels*, Algorithmica **47** (2007), pp. 183–201.

[3] Chlebik M., J. Chlebikova, *Inapproximability results for bounded variants of optimization problems* - Electronic Colloquium on Computational Complexity, Report No. 26, (2003).

[4] Clementi A. E. F., P. Penna and R. Silvestri, *On the Power Assignment Problem in Radio Networks*, Electronic Colloquium on Computational Complexity (ECCC), Technical Report 054 (2000).

[5] Erzin A.I., R. V. Plotnikov and Yu. V. Shamardin *On some polynomially solvable cases and approximate algorithms in the optimal communication tree construction problem*, Journal of Applied and Industrial Mathematics **7** (2013), pp. 142–152.

[6] Fuchs B., *On the Hardness of Range Assignment Problems*, Electronic Colloquium on Computational Complexity, Report No. 113 (2005).

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

[8] Hansen P. and N. Mladenović, *Variable Neighborhood Search Methods*, Les Cahiers du GERAD, G–2007–52, 2007.

[9] Kirousis L. M., E. Kranakis, D. Krizanc and A. Pelc., *Power consumption in packet radio networks*, Theoretical Computer Science **243** (2000), pp. 289–305.

[10] Zalyubovskiy V., A. Erzin, S. Astrakov and H. Choo, *Energy-efficient Area Coverage by Sensors with Adjustable Ranges*, Sensors **9** (2009), pp. 2446–2460.