Location and Design of Ground Stations for Software Defined Satellite Networks

1st Anton Eremeev
Sobolev Institute of Mathematics
SB RAS
Omsk, Russia
eremeevtmp@yandex.ru

2nd Alexandra Gette
Dostoevsky Omsk State University
Omsk, Russia
gette4aleks@yandex.ru

3rd Sergei Hrushev
Sobolev Institute of Mathematics
SB RAS
Omsk, Russia
hrushev@omsknet.pro

4th Tatyana Levanova
Sobolev Institute of Mathematics
SB RAS
Omsk, Russia
levanovat@yandex.ru

Abstract—**In this paper, the problem of locaion ground stations in a global satellite communications network is formulated and a variable neighborhood search algorithm (VNS) for its solving is proposed. In this problem, it is required to place and configure the design of the ground stations, so as to maximize the number of delivered data packets, given a budget constraint and the set of sessions between the users or the users and the Internet. The overall amount of delivered packets for any tentative location and design of ground stations is estimated by means of approximate solving of the length-bounded fractional maximum multiproduct flow problem where all edges are of length one. Results of computational experiments and comparison of VNS with local search algorithms are provided.**

Keywords—routing problem, location, software defined network, greedy algorithm, FPTAS, computational experiment

I. Introduction

In this article, we formulate and propose a method for solving the problem of locating and designing ground stations in a global satellite communications network. We assume that the network consists of satellites in low Earth orbit, ground stations (providing Internet access) and a network operations control center (NOCC) (see e.g. [1, 2]). At each potential location of a ground station, such a station can be created according to one of the possible options or not created at all. The design of a ground station is described by the throughput of connections to satellites, the throughput of connections with other ground stations via ground Internet network, and the cost of using this design of a ground station per unit of time. It is assumed that the system operates in discrete time intervals, which are small enough to assume that the quality of communication channels and customer requirements do not change over one time period. The data packet routes for each source-destination pair are calculated in the Network Operations Control Center.

The packet routes for each pair of source and destination are calculated in NOCC in real time, and each node (satellite or ground station) receives the routes for all packets originating from this node. Each packet sent from source to a destination contains some data content and the packet's route. A maximal admissible number of edges L in packet paths is imposed due to a technical limitation on the number of bits reserved for encoding a packet path. Short packet paths also typically have low latency. For the sake of simplicity, we assume that each task instance describes the system in one time interval, and all requirements for this time interval are known in advance.

The solution to the problem of placement and design of ground stations can be divided into two stages: 1) placement of ground stations and selection of design configuration for each of them; 2) routing of information packets between clients within a certain time interval. The decision variables of the first stage determine the set of ground stations under construction and their design options. At the second stage, packet routes are calculated in each time interval, taking into account session requests from the clients and the current state of the communication network. Quality of service (QoS) in a time interval is measured as the average number of lost or non-routed packets per unit of time. i.e., the difference between the number of packets for the requested sessions and the number of routed packets. Therefore, the number of routed packets is considered as the maximization criterion in what follows. The total budget for the use of ground stations over a given period of time is imposed as a constraint.

In preceding publications, different approaches to solving the packets routing problem have been studied theoretically and experimentally [3] and the graph-based models of network with different levels of detail have been discussed [4].

The novelty of this work lies in the fact that an algorithm for searching for a variable neighborhood (VNS) is proposed to solve the problem of locating and designing ground stations. The algorithm is tested on examples of synthetic data motivated by one of the promising satellite communication networks. An experimental comparison of VNC with some simple local search algorithms is given.

II. Problem of Location and Design of Ground Stations for Software Defined Satellite Networks

A. Packet Routing Problem Formulation

In the packet routing problem, we are given the following as the input:

$G = (V, E)$ is the digraph where V is the set of the satellites and ground stations, and E is the set of connections (channels);

$n=|V|$, $m=|E|$;

$u(e) \geq 0$ is the throughput of connection e;

N is the total number of active sessions in the system.

Each session i is represented by a triple (A_i, B_i, W_i), $i=1, \ldots, N$, where $A_i, B_i \in V$ is a pair of source-destination nodes, W_i is data traffic per time unit during the session. Each session is considered here as a session between the nodes of graph G (ground stations or satellites) to which the users are currently connected.

L is the maximal admissible number of edges in a packet path.

$\tau(e)$ is a delay in data transmission over a channel $e \in E$.

The problem asks to find a set of paths in G for so that each session corresponds to at most one route, and the maximum possible number of sessions is routed this way. The problem has constraints thet the maximum number of arcs in each path does not exceed L and that for each arc $e \in E$, the total amount of information transmitted over all routes passing through e must not exceed the capacity $u(e)$. As a secondary criterion, we consider the maximum delay of the routed sessions.

Note that a formulation of the problem would be NP-hard if we introduced an upper bound of the secondary criterion. This is due to the NP hardness of the fractional length-bounded maximum multicommodity flow [5]. If the packets routing problem must be solved quickly in real time, the problem is formulated only for maximization of the main criterion. The secondary criterion is taken into account only in the greedy-type heuristic below.

B. *Formulation of the Problem of Placement and Design of Graund Stations*

At each potential location of a ground station, such a station can be created according to one of the possible options or not created at all. The design of the ground station is characterized by:

- throughput of connections with satellites;
- throughput of connections with other ground stations;
- the cost of using this station per unit of time.

Let's introduce the notation:

I is the set of locations for ground stations.

B is the admissible budget to be spent on ground stations construction and utilization.

R is the set of configurations (design variants), possible for each ground station.

c_i^r is the cost of opening a configuration of type r at location i.

z_i^r is equal to 1 if in location i the station with configuration r is placed, and 0 otherwise.

$f(Z)$ is the solution to the packet routing problem, given the vector of configurations $Z=(z_i^r)$.

Constraints: Σ

1) $\sum_{i \in I} \sum_{r \in K} c_i^r z_i^r \leq B$, the budget constraint;
2) $\sum_{r \in K} z_i^r \leq 1$, no more than one configuration may be chosen at each location;
3) $\sum_{i \in I} \sum_{r \in K} z_i^r \geq 1$, at least one ground station is open.

It is required to place ground stations so that the total number of routed packets is maximal.

This work was supported by the RSF, №21-41-09017.

III. Greedy Algorithm for the Packet Routing Problem

As it was noted in [4], if the transmission rate of each session is can be considered negligible compared to the channel bandwidth and source-to-destination transmission requirements then the packet routing problem may be approximated by the fractional length-bounded maximum multicommodity flow problem with unit edge weights and k different commodities (the source-destination pairs of sessions in our case). The latter may be solved exactly using an LP formulation, which involves $O(Lkn+m)$ constraints and $O(Lkm)$ variables [3,6] or approximately with any required relative error, using the fully polynomial time approximation scheme [3]. However, these approaches require significant amount of CPU time, so that evaluation of numerous tentative location and design options for the ground stations within a metaheuristic algorithm becomes practically impossible. In order to overcome this issue, in what follows we will use a relatively fast greedy heuristic from [4].

The basic principle of the greedy algorithm [4] is to assign sessions to routes iteratively, accepting only routes of at most L edges. After routing a session (A_i, B_i, W_i), the capacity of edges along the shortest path from A_i to B_i is reduced by W_i. Whenever the capacity of an arc is exhausted, this arc is excluded from further consideration. The session assignment sequence in the greedy algorithm is by descending shortest path delay. This rule was chosen to reduce the maximum latency and to satisfy the maximum arcs per path constraint L, since sessions with the highest latency on the shortest paths are more likely to violate the upper bound L.

The shortest paths between all pairs of vertices one can use the algorithms of Dijkstra's, Floyd-Warshall or Bellman-Ford algorithm. Because the set of shortest paths is evaluated at most m times, the time complexity of the greedy algorithm is $O(mn^3)$ if Floyd-Warshall algorithm is used. The Bellman-Ford algorithm can be easily truncated so that the returned set of paths contains no more than L edges in each path. Then, if n parallel processors are available, the greedy algorithm may be implemented in $O(m^2 L)$ time. This greedy algorithm has no approximation guarantee, but in the experimental testing in [4] it has shown a relative error at most 7% on the problem instances of the same type as we use in the present paper.

IV. Variable Neighborhood Search

The variable neighborhood search (VNS) was proposed in [7]. It is based on the following idea: A local optimum in one neighborhood may not be a local optimum in another neighborhood. If we look at all possible neighborhoods, then one of the local optima will be the global optimum. Then, by changing the neighborhood, you can continue the search and find a better solution. Mladenovich and Hansen were the first to suggest working with variable neighborhoods. They presented a basic outline of the VNS approach and illustrated its effectiveness for the Traveling Salesman problem. VNS has been used to solve various object placement problems (p-median problem, Weber problem, competitive placement problem, etc.). In the present paper, we adapt the VNS algorithm to the problem of location and design of ground stations.

Outline of the Variable Neighborhood Search

Suppose that for each solution Z, a subset $N(Z)$ in the space of feasible solutions is defined (a neighborhood of Z).

The family of all sets *N(Z)* is called a neighborhood system. In what follows, we use the basic scheme of the VNS. Suppose there are *K* systems of neighborhoods N_k, *k=1, …,K,* which may be used in the VNS.

Initialization: choose the initial feasible solution *Z* at random.

```
Repeat the following steps until stopping criterion is met.
Set $k:=1$.
While $k < K+1$, repeat:
    (a) Generate a point $Z'$ randomly from $N_k(Z)$;
    (b) Apply local search from the starting point $Z'$; let $Z''$ denote the local optimum found by the local search;
    (c) If the local optimum $Z''$ improves the current solution $Z$, then move there: $Z:=Z''$, $k:=1$, and continue searching; otherwise move to next neighborhood, that is, set $k:=k+1$.
Return $Z$.
```

As a stopping criterion, the maximum computation time is used. If the VNS looked through the entire list of neighborhoods in less time, then it restarted.

A. Neighborhoods

An important step in the development of the algorithm is to determine the type of neighborhoods. We used three neighborhoods.

In the first neighborhood (denoted N_1), an admissible solution is called a neighbor if it differs from the given solution by 1 in one of the components.

In the second neighborhood (denoted N_2), an admissible solution is called a neighbor if it differs from the given solution by –1 in of the components.

In the third neighborhood (denoted N_3), a solution is called a neighbor if it differs from the given solution by 1 or –1 in one of the components.

V. Computatinal Experiment

The algorithm is implemented in Python and tested using Intel Xeon CPU X5675. Testing instances were built on the basis of a series of instances of packet routing problem in global satellite networks from [4].

The pairs of source and destination nodes were generated to roughly model the communication flows on the global scale. It was assumed that the number of users is proportional to the human population. The source and destination nodes of each session were randomly generated users. The users were assigned to the nearest ground station or satellite.

It was assumed that the channel bandwidth is between 100 and 300 Mbps. The transmission rate of each session was assumed to be 9600 bps.

The computational experiment was carried out on 6 test instances, in each of these examples the overall number of transmitted packets was 2355465600.

The expert location of the ground stations, followed by the greedy packet routing algorithm yielded the results presented in Table I.

TABLE I. Average number of Lost Packets (EXPERT SOLUTION)

problem 1	problem 2	problem 3	problem 4	problem 5	problem 6
$2435 \cdot 10^5$	$2235 \cdot 10^5$	$2219 \cdot 10^5$	$2477 \cdot 10^5$	$2220 \cdot 10^5$	$2237 \cdot 10^5$

For each instance, the VNS algorithm worked for the same ·amount of time, namely 5, 10, 15, 30, 60 minutes. Such a duration is considered acceptable from a practical point of view. The results of the VNS algorithm are presented in Table II.

TABLE II. Number of Lost Packets (VNS results)

min	problem 1	problem 2	problem 3	problem 4	problem 5	problem 6
5	$2435 \cdot 10^5$	$2310 \cdot 10^5$	$2144 \cdot 10^5$	$2312 \cdot 10^5$	$2230 \cdot 10^5$	$2254 \cdot 10^5$
10	$2448 \cdot 10^5$	$2218 \cdot 10^5$	$2084 \cdot 10^5$	$2468 \cdot 10^5$	$2070 \cdot 10^5$	$2179 \cdot 10^5$
15	$909 \cdot 10^5$	$1647 \cdot 10^5$	$1846 \cdot 10^5$	$1273 \cdot 10^5$	$1616 \cdot 10^5$	$1866 \cdot 10^5$
30	$1678 \cdot 10^5$	$1294 \cdot 10^5$	$1584 \cdot 10^5$	$800 \cdot 10^5$	$1003 \cdot 10^5$	$966 \cdot 10^5$
60	$1499 \cdot 10^5$	$847 \cdot 10^5$	$700 \cdot 10^5$	$665 \cdot 10^5$	$864 \cdot 10^5$	$865 \cdot 10^5$

In addition to the VNS algorithm, the local search method was applied using a single neighborhood N_1, N_2 or N_3 separately. Algorithm N1 uses only first neighborhood, algorithms N2 and N3 use the second and third neighborhoods, respectively. In the algorithm "N1,N2,N3", these three neighborhoods are looked through sequentially. All algorithms were given the same CPU time. The results can be seen in Fig.1-3.

Within the counting time of 5 and 10 minutes, all algorithms showed similar results. Therefore in what follows we provide the graphs of the obtained results when the CPU time is equal to 15, 30 and 60 minutes. Figs. 1-3 show that with increasing time, the advantage of VNS over other considered variants of local search algorithms increases.

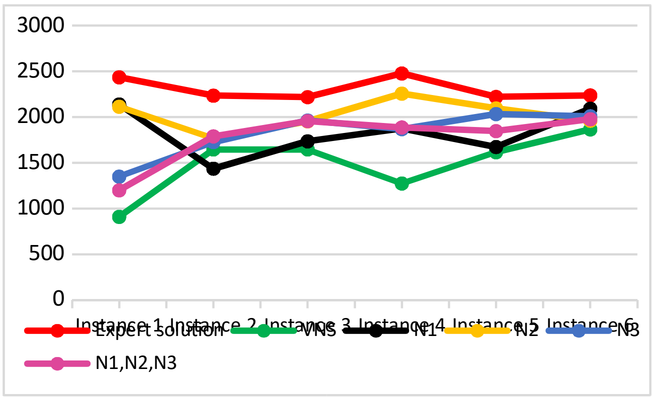

Fig. 1. Average number of lost packets for 15 min run.

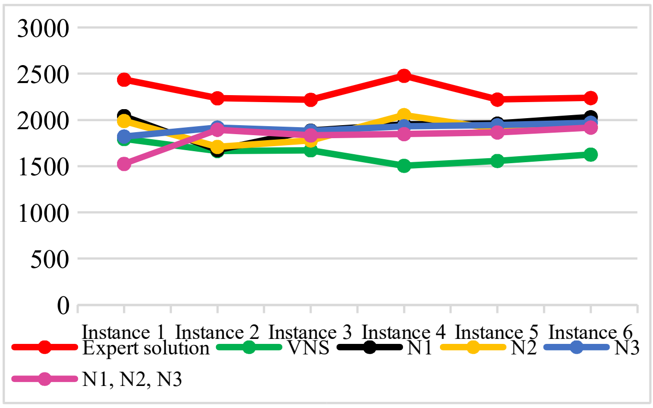

Fig. 2. Average number of lost packets for 30 min run.

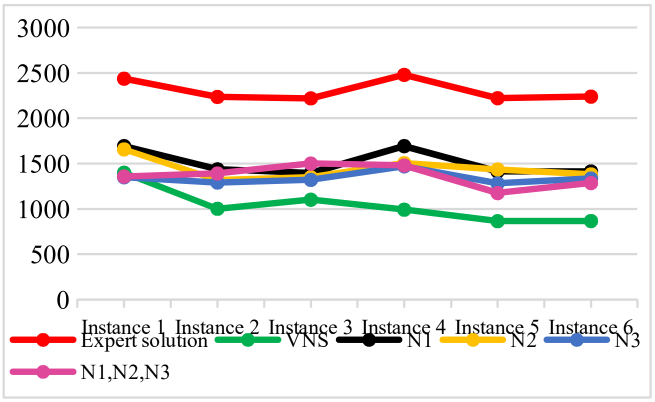

Fig. 3. Average number of lost packets for 60 min run.

VI. Conclusions

The obtained results indicate that VNS is a promising method for approximate solution of the ground stations location and design problem. In this study, on the second stage we considerate only a single period of network functioning and assumed that the fractional maximum multicommodity flow model provides an adequate approximation of the system within a single period. We expect that in the future research, this model will be refined by a simulation model, and a simulation-optimization approach will be developed for its solution.

[1] S. Xu, X.W. Wang, and M. Huang, "Software defined next-generation satellite networks: Architecture, challenges, and solutions," IEEE Access, vol. 6, 2018, pp. 4027–4041.

[2] Z. Tang, B. Zhao, W. Yu, Z. Feng, and C. Wu, "Software defined satellite networks: Benefits and challenges," 2014 IEEE Computers, Communications and IT Applications Conference, pp. 127–132.

[3] P. Borisovsky, A. Eremeev, S. Hrushev, V. Teplyakov, and M. Vorozhtsov, "On three approaches to length-bounded maximum multicommodity flow with unit edge-lengths," Yugosl. Journ. of Oper. Res., vol. 29, N 1, 2019, pp. 93–112.

[4] P. Borisovsky, A. Eremeev, S. Hrushev, and V. Teplyakov, "Experimental evaluation of algorithms for packet routing in software defined network," ScienceDirect IFAC PapersOnLine 55–10 (2022).

[5] G. Baier, "Flows with Path Restrictions," Ph.D. Dissertation, TU Berlin, Berlin, 2003.

[6] P. Kolman and C. Scheideler, "Improved bounds for the unsplittable flow problem," J. Algorithms, vol. 61, no. 1, pp. 20–44, 2006.

[7] N. Mladenovic, and P. Hansen, (1997) "Variable Neighborhood Search," Computers and Operations Research, vol. 24, 1997, pp. 1097–1100.