Paper

Greedy randomised adaptive search procedures for topological design of MPLS networks

Andrzej Mysłek

Abstract — In this paper, the IP/MPLS network cost optimisation problem of selecting localisation of nodes and links, combined with link's dimensioning, is discussed. As the considered problem is hard, we discuss and propose greedy randomised adaptive search procedure (GRASP) based solution method. GRASP is an iterative randomised sampling technique which combines adaptive randomised greedy function in constructing initial solution with local search optimisation. The effectiveness of the method is illustrated by means of a numerical study. We compare the GRASP results with results for both exact and heuristic methods obtained in previous research concerning topological design problem.

Keywords — network design, optimisation, MPLS, GRASP, local search.

1. Introduction

New communications systems, such as ATM or IP backbone networks, have to be deployed in a short time due to user's demand for provision of new services. Furthermore, ensuring the high quality of supplied services is required. Whereas the IP network does not possess built-in capabilities to ensure the quality of service (QoS), the multiprotocol label switching (MPLS) introduced to the IP network can add mechanisms for QoS assurance.

Effective and fast network deployment requires good network design tools, incorporating fast network optimisation algorithms for diverse optimisation tasks. To identify the problem described in this paper in the range of different network optimisation problems, we can use the following classification, proposed in [1]:

- **access network topology optimisation** problems are usually looking for hierarchical network structures, assuming a simple traffic demand pattern;
- **backbone network topology optimisation** problems require (arbitrary) mesh topology with traffic demands between each pair of end nodes given.

The optimisation problem, called the transit node and link localisation problem (TNLLP), is a general instance of the backbone network topology optimisation problem.

The TNLLP is characterised as follows.

For a given set of access nodes and the demand between each pair of access nodes find:

- number and locations of the actually installed transit nodes (in these nodes non traffic is originated, they only switch the traffic streams between access node pairs);
- capacity of links connecting access nodes to transit nodes;
- capacity of links interconnecting transit nodes;

such that the demand is realised at a minimum total network cost, which is composed of:

- fixed installation cost of each transit node,
- fixed installation cost of each link,
- capacity-dependent cost of each link.

The considered problem can be applied to MPLS-capable IP networks, with the access nodes representing ingress/egress label edge routers (LER), and the transit nodes – label switching routers (LSR). Since the evolving IP/MPLS networks will comprise large numbers of LERs and LSRs to be placed in many possible sites, the problem of optimal node and link location becomes important, especially for economical network extension. The problem definition is general enough to describe topology design problem without node localisation (cf. ONDP in [1]).

A subproblem of the TNLLP, known as optimal network design problem (ONDP) was widely studied in [2–5] using greedy and branch-and-bound algorithms. In [6] the multicommodity capacitated network design problem, an arc-based formulation with variable cost of flow and installation cost of multiple facilities installed on the arc, is solved combining the cutting plane method with the Lagrangean relaxation and heuristics. In Ref. [7] the dualascent procedure in the Lagrangean relaxation to solve an uncapacitated network design problem is used. Other references can be found in [1].

Since a problem very similar to the considered one is known to be NP-complete [8], one cannot expect to find time efficient algorithms for exact solving of the TNLLP. The branch-and-bound approach [9], although exact, is in general too time-consuming, especially for large networks. Hence, we have to use heuristic methods. Some heuristics were already studied in [1], and some of them (H4B, SAL, SAN) supply quite good suboptimal solutions. In this paper we inspect another heuristic method called the greedy randomised adaptive search procedure, that comprises greedy,

but randomised construction of the initial solution, and the local search method. GRASP has already been applied to many different problems (cf. [10–12]), and seems to be a good metaheuristic method, which may be adapted to solve TNLLP.

This paper is a continuation of [1] and [9], and it is organised as follows. In Section 2 we give a formal statement of the considered problem. Section 3 consists of GRASP method's presentation. A discussion of GRASP implementation for TNLLP is provided in Section 4. In Section 5 we illustrate efficiency of the GRASP implementation using numerical examples. Conclusions are drawn in Section 6.

2. Problem formulation

Let us consider two disjoint sets of nodes: access nodes labelled with $w = 1,\ldots,W$, and transit nodes labelled with $v = 1,\ldots,V$. The nodes are connected by undirected links, labelled with $e = 1,\ldots,E$. The links connecting access nodes to transit nodes are called access links, and transit links interconnect transit nodes. There are no links between the access nodes. The incidence of links and transit nodes is given by the binary incidence coefficients $b_{ev} : b_{ev} = 1$ if link e is incident with transit node v, and $b_{ev} = 0$ otherwise.

The access nodes are installed (fixed) and cost nothing[1]. A transit node v can be provided or not. If it is installed, it costs l_v. Any transit or access link can also be provided or not. An installed link costs $c_e y_e + k_e$ (where y_e is the capacity of the link e), otherwise it costs nothing. Since the network must be consistent, if a link is provided then its end nodes must be provided as well.

The demands, which are imposed on network and labelled with $d = 1,\ldots,D$, are realised by means of flows allocated to admissible paths. With each demand d there is associated its origin node s_d, destination (target) node t_d and demand volume h_d.

There are two main ways of formulating the flow problem for such a graph: link-path formulation and node-link formulation. The former requires to define for each demand a set of admissible paths labelled with $j = 1,\ldots,J_d$. Each such path begins at node s_d, then traverses (a non-empty) subset of transit nodes, and ends at node t_d. The paths are defined by the binary incidence coefficients a_{edj}, where $a_{edj} = 1$ if link e belongs to path j of demand d, and $a_{edj} = 0$ otherwise.

Transit nodes and links localisation problem can be formally stated in its link-path formulation as the following mixed-integer programme (MIP).

[1]Introduction of the installation cost of the access nodes would change an objective function just for a constant value, because provision of access links is determined by imposed demands. Hence, it does not change our optimisation problem.

TNLLP1 (link-path formulation)

indices

$d = 1,2,\ldots,D$	demands
$j = 1,2,\ldots,J_d$	paths for realising flows of demand d
$v = 1,2,\ldots,V$	transit nodes
$e = 1,2,\ldots,E$	links

constants

h_d	volume of demand d
a_{edj}	$= 1$ if link e belongs to path j of demand d, 0 otherwise
c_e	cost of capacity unit of link e
k_e	fixed cost of installing link e
Y_e	upper bound for the capacity of link e
b_{ev}	$= 1$ if link e is incident with transit node v, 0 otherwise
l_v	fixed cost of installing transit node v
Y_e	upper bound for the capacity of link e
G_v	upper bound for the degree of transit node v

variables

x_{dj}	flow realising demand d allocated to path j (non-negative continuous variable)
y_e	capacity of link e (non-negative continuous variable)
σ_e	$= 1$ if link e is provided, 0 otherwise (binary variable)
ε_v	$= 1$ if node v is provided, 0 otherwise (binary variable)

objective

$$\text{minimise} \quad C = \Sigma_e c_e y_e + \Sigma_e k_e \sigma_e + \Sigma_v l_v \varepsilon_v \tag{1}$$

constraints

$$\Sigma_j x_{dj} = h_d \quad d = 1,2,\ldots,D \tag{2}$$

$$\Sigma_d \Sigma_j a_{edj} x_{dj} = y_e \quad e = 1,2,\ldots,E \tag{3}$$

$$y_e \le Y_e \sigma_e \quad e = 1,2,\ldots,E \tag{4}$$

$$\Sigma_e b_{ev} \sigma_e \le G_v \varepsilon_v \quad v = 1,2,\ldots,V. \tag{5}$$

In TNLLP1 constraints (2) assure that the demands are realised, and constraints (3) and (4) that the capacity of a non-provided link is equal to 0. Constraints (5) assure that if a link is provided then also its end nodes are. The objective is to minimise the cost of all links and of transit nodes. Parameters Y_e and G_v are high enough not to limit the capacity of edges and the number of edges outgoing from node respectively.

The set of admissible paths for each demand d may be defined as the set of all paths between s_d and t_d, or it may be somehow limited (e.g. to at most two-hop paths). In the former case, the values of J_d can be very large, leading to an excessive amount of flow variables in the link-path problem formulation given above. For such a problem, we can use the alternative node-link problem formulation.

The node-link formulation of TNLLP uses link flows instead of path flows and requires the flows to be directed. Hence, with each link e we associate two directed arcs traversing the link in two opposite directions. Note that also the demands are directed. To suppress the number of flow variables we use flow variables aggregated by originating node (non-negative continuous variables).

TNLLP2 (node-link formulation)
indices

$v = 1,2,\ldots,V$ transit nodes
$w = 1,2,\ldots,W$ access nodes
$e = 1,2,\ldots,E$ links
$t = 1,2,\ldots,T$ directed transit arcs (between transit nodes)
$f = 1,2,\ldots,F$ directed access arcs (between access and transit nodes)

constants

$h_{ww'}$ demand originating at access node w and destined for access node w
H_w $= \Sigma_{w'} h_{ww'}$ total demand originating at access node w
b_{ev} $= 1$ if node v is incident with link e, 0 otherwise
b_{tv} $= -1$ if transit arc t is incoming to transit node v
$= 1$ if transit arc t is outgoing from transit node v
$= 0$ otherwise
b_{fv} $= -1$ if access link f is incoming to transit node v
$= 1$ if access link f is outgoing from transit node v
$= 0$ otherwise
b_{fw} $= -1$ if access link f is incoming to access node w
$= 1$ if access link f is outgoing from access node w
$= 0$ otherwise
a_{et} $= 1$ if transit arc t is realised on link e, 0 otherwise
a_{ef} $= 1$ if access arc f is realised on link e, 0 otherwise
Y_e upper bound for the capacity of link e
G_v upper bound for the degree of transit node v

variables

x_{tw} flow realising all demands originating at access node w on transit arc t
x_{fw} flow realising all demands originating at access node w on access arc f
y_e capacity of link e
σ_e $= 1$ if link e is installed, 0 otherwise (binary variable)
ε_v $= 1$ if node v is installed, 0 otherwise (binary variable)

objective

minimise (1)

constraints

$$\Sigma_t a_{et} \Sigma_w x_{tw} + \Sigma_f a_{ef} \Sigma_w x_{fw} = y_e \quad e = 1,2,\ldots,E \tag{6}$$

$$\Sigma_f b_{fw'} x_{fw} = H_w \quad w = 1,2,\ldots,W \tag{7}$$

$$\Sigma_f b_{fw'} x_{fw} = h_{ww'} \quad w = 1,2,\ldots,W, \quad w' = 1,2,\ldots,W \tag{8}$$

$$\Sigma_t b_{tv} x_{tw} + \Sigma_f b_{fv} x_{fw} = 0 \quad v = 1,2,\ldots,V, \quad w = 1,2,\ldots,W \tag{9}$$

$$y_e \le Y_e \sigma_e \quad e = 1,2,\ldots,E \tag{10}$$

$$\Sigma_e b_{ev} \sigma_e \le G_v \varepsilon_v \quad v = 1,2,\ldots,V. \tag{11}$$

In the formulation of the constraints in TNLLP2 only the following flow variables are used:

- x_{tw} for all pairs (t,w) such that $t = 1,2,\ldots,T$, $w = 1,2,\ldots,W$
- x_{fw} for all pairs (f,w) such that $f = 1,2,\ldots,F$, $w = 1,2,\ldots,W$ and either access link f is outgoing from access node w (i.e. $b_{fw} = 1$) or link f is incoming to some other access node w' with $h_{ww'} > 0$ ($b_{fw} = -1$ and $h_{ww'} > 0$).

In TNLLP2 each link e supports two directed arcs (either both access or both transit) whose numbers are specified by coefficients a_{ef} and a_{et} (respectively) equal to 1. Constraint (7) forces the total demand H_w generated in access node w to flow out, constraint (8) – that the portion $h_{ww'}$ of the flow originated at node w and destined for node w' stays at w', and constraint (9) – that no flow stays in a transit node.

The savings in the number of variables and constraints while using TNLLP2 instead of TNLLP1 are illustrated in [1].

Note that in the optimal solution of TNLLP, the demands can be routed on single paths, so if a demand corresponds – to an MPLS tunnel, the solution guarantees that the tunnel is realised on one path. This follows from the linearity of the capacity-dependent cost used in the cost function: for each fixed configuration of nodes and links, the capacity dependent part of (1) is minimised by assigning the entire demand volume h_d to one of its shortest paths with respect to the link metrics c_e. This property was used in [9] to construct a lower bound for the branch-and-bound algorithm.

3. Greedy randomised adaptive search procedure

A greedy randomised adaptive search procedure [10, 11] is one of neighbourhood search methods, like local search and tabu search methods. The GRASP is metaheuristic approach, that was applied to various optimisation problems: graph, scheduling, assignment and other problems, and for different domain-specific problems like aircraft routing or network planning (for the full bibliography of the GRASP see [12]).

A general GRASP heuristic is a two-phase iterative process. In each GRASP iteration there are two phases – the construction of a new greedy randomised solution, which consequently is used as the starting point for the second phase, and a local search algorithm that is run afterwards. This procedure is executed repeatedly until some termination criterion is met. The best solution over all iterations is the result. Pseudo-code for a generic GRASP method is given in Source 1.

Source 1. Pseudo-code for GRASP

```
procedure GRASP (var bestSolution);
begin
  repeat
     ConstructGreedyRandomisedSolution(solution);
     LocalSearch(solution);
     UpdateBestSolution(solution,bestSolution);
  until TerminationCriterion();
end;
```

In the construction phase we are looking for a solution, building it up from smaller parts. If we describe the solution as a vector of, for example, binary decision variables, constructing a solution means to choose variables one by one, and decide about their values. Randomly choosing and setting one of the best variables is the probabilistic component of the GRASP method.

To make construction process more formal and applicable to different problems, in Ref. [11], the concept of a list of candidates, called restricted candidate list (RCL) is introduced. The RCL is a list of the best candidates (parts of a solution) that can be used to form a new solution. It is restricted, because instead of looking for just the best candidate we allow worse candidates to be chosen to form the solution. Candidates are chosen randomly, and after adding a candidate to the solution, we adapt greedy function, i.e. we calculate the impact of our choice for the greedy function. This procedure is presented in Source 2.

Source 2. Pseudo-code for the GRASP construction phase

```
procedure ConstructGreedyRandomisedSolution
  (var solution);
begin
  solution = Ø;
  while not SolutionConstructed(solution) do
  begin
    MakeRCL(RCL);
    s = SelectRandom(RCL);
    solution = solution ∪ {s};
    AdaptGreedyFunction(s);
  end;
end;
```

The solution obtained in the construction phase is not guaranteed to be locally optimal with respect to simple neighbourhood definition. Hence, we try to improve the solution, applying a local search loop. This algorithm works in an iterative fashion by moving to a better solution in the neighbourhood of the current solution. It terminates when no better solution can be found (in the neighbourhood). A neighbourhood $N(s)$ of the solution s relates this solution to a set of solutions. Choosing the best solution from the neighbourhood we run the greedy local search.

The two phases of GRASP give foundation for the good optimisation technique. A greedy choice of candidates forms random (but good) solution, which is then improved by a local search.

4. GRASP implementation for TNLLP

Greedy randomised adaptive search procedure uses a two-phase iterative approach to solve the problem. In the first phase, the construction phase, construction of restricted candidate list is repeated. In each iteration of the construction phase we choose a part of the solution to form initial solution for the second phase of GRASP.

In TNLLP a solution is a set of edges and nodes provided, and the set of paths used to realise demands (each demand on a single path). Note, that if we decide which edge and node should be provided, we can easily find paths for demands using the shortest path according to variable cost of edges. Hence, our construction phase would consist of repeatedly choosing one edge and adding it to the solution. We could select, for example, edges with the highest flow belonging to the shortest path according to the modified weights from [9] (incorporating the installation cost of nodes and links, and the variable cost of edges).

Selecting edges to form a solution seems to be a good method. We can rank edges according to their flow or the number of demands realised on and we have termination criterion for construction (when all the edges not chosen do not bare the flow). However, the solutions generated by this method are usually poor.

The better starting solution can be constructed using demand's flows allocation. This method was used in [1] to construct starting solution for heuristics, but here we can add some randomisation. Let each demand's flow be routed through the shortest paths, according to the current state of the network. We calculate the length of the path as the sum of the variable cost of the demand and the fixed cost of not provided links (and optionally nodes) on that path. We adapt our greedy function by setting the edges (and nodes) as provided after the shortest path is found. Our modified construction phase is shown in Source 3.

Source 3. Pseudo-code for the GRASP modified construction phase (for TNLLP)

```
procedure ConstructGreedyRandomisedSolution
  (var solution);
begin
  solution* = Ø;
  MakeRCL(demandRCL);
  repeat
    d = SelectRandom(demandRCL);
    path = FindShortestPath(solution,d);
    solution = solution ∪ NodesAndEdges(path);
  until size(demandRCL)= 0;
end;
```

* solution consists of edges and nodes

In the construction phase we first construct a demand RCL. Then, selecting in each iteration one demand at random, we route this demand through the shortest path, updating the solution. When all the demands are routed, the construction phase is done. Our initial solution is good enough to apply efficiently the local search phase. Moreover, our initial solutions are well randomised, since the order of demands routed is random in each run of the construction phase.

Next, we try to improve GRASP initial solution obtained in the construction phase applying a local search. To identify the local search procedure for TNLLP we have to choose the neighbourhood model (the solution model was determined by the construction phase). Certainly, we could use the model from the construction phase (the demand reallocation), but to make the search more exhaustive we select nodes and links switching. An iteration of the local search will comprise switching selected edge (or node), i.e. making edge unavailable if it is provided and vice versa.

Switching an edge off requires rerouting demands which use the edge. It is rather simple, since the shortest paths of other demands do not change and there are not so many edges provided. Alternatively, switching edge on should involve rerouting all the demands, because any shortest path can change[2].

Finally, a GRASP iteration is composed of the construction phase, where we obtain the initial solution randomly routing demands on the shortest paths, and the local search, that improves the initial solution. We repeat the GRASP iteration until some arbitrary termination criterion is met, for example maximum iteration limit is reached or there was no improvement for last n iterations.

The GRASP for TNLLP, as it was described above, has some variants. We can use the installation cost of transit nodes during construction phase combined with the variable cost of demand and installation cost of edges, or just the variable cost of demand and installation cost of edges to determine the shortest path for a demand. In the local search phase the nodes can be switched first, and then edges instead of switching edges only.

5. Numerical results

We have considered three artificially generated network structures (N7, N14 and N28) determined by the geographical locations of nodes, and one realistic network (PL49) reflecting Polish public backbone network. The networks were used in [1] and are available on the web site [13]. The basic parameters of the networks are given in Table 1. All links are potentially available.

The unit cost c_e of link e is in all cases proportional to its geographical length. The fixed installation cost is given by $k_e = c_e \cdot 10^n$ (n is a parameter in computations; the fixed cost of access links is additionally multiplied by 3, and for the transit links by 2). The fixed installation cost of a transit node is the same for all nodes and is given by $l_v = 10^k$ (k is another parameter in computations).

[2]Finding the shortest path is the most time-consuming activity in the demand's routing.

Table 1
Test networks parameters

Network	W	V	F	T	D	$\min h_d$	$\max h_d$	$\Sigma_d h_d$
N7	7	5	70	20	42	240	1920	34 320
N14	14	11	308	110	182	120	7560	172 320
N28	28	15	840	210	756	120	30 240	892 492
PL49	49	12	1178	132	2352	36	53 572	2 788 073

Applied GRASP methods vary in network installation cost factors used in the solution construction, node switching and edge adding. G1 operates on the fixed cost of nodes and the cost of flow (variable cost of edges times demand volume) during the construction phase and removes edges in the local search. The fixed cost of nodes was added as a factor in the construction phase of G2. Methods G3 and G4 are respectively similar to G1 and G2, but in a local search phase they try to switch off transit nodes first, and then remove edges. All the four methods G1–G4 have their counterparts G1$^+$–G4$^+$, which in the local search phase not only remove edges, but can add them too.

Table 2
Results for TNLLP (cost in units of 10^6)

Network	n	k	H4B*	SAN*	SAL*	EX*	G1	G2
N14	4	4	58.93	81.38	57.61	57.61	58.54	58.54
N14	4	5	59.92	81.83	58.60	58.60	59.53	62.24
N14	4	6	69.82	84.07	69.53	67.14	69.43	75.28
N14	5	4	266.23	426.88	265.32	—	276.39	276.39
N14	5	5	267.22	427.24	267.22	—	277.38	277.38
N14	5	6	277.12	430.84	277.88	—	287.28	287.28
N28	4	4	253.21	274.45	262.46	—	262.32	262.32
N28	4	5	254.56	277.21	256.34	—	263.67	263.67
N28	4	6	268.06	283.51	284.04	—	277.17	318.72
N28	5	4	751.60	1369.97	742.94	—	781.85	781.85
N28	5	5	752.95	1370.78	769.64	—	783.20	795.81
N28	5	6	761.37	1378.88	784.93	—	796.70	809.31
PL49	4	7	1199.75	1186.80	1207.50	—	1272.84	1361.90
* Results for the best methods published in [1].								

Table 3
Results for TNLLP – continued (cost in units of 10^6)

Network	n	k	G3	G4	G1$^+$	G2$^+$	G3$^+$	G4$^+$
N14	4	4	58.54	58.54	57.61	57.61	57.61	57.61
N14	4	5	59.53	62.24	58.60	58.60	58.60	58.60
N14	4	6	69.43	75.28	68.50	75.28	68.50	75.28
N14	5	4	276.39	276.39	271.25	271.25	271.25	271.25
N14	5	5	277.38	277.38	272.24	272.24	272.24	272.24
N14	5	6	287.28	287.28	282.14	282.14	282.14	282.14
N28	4	4	262.32	262.32	250.02	250.02	250.02	250.02
N28	4	5	263.67	263.67	251.37	251.86	251.37	251.86
N28	4	6	277.17	318.72	264.87	277.15	264.87	277.15
N28	5	4	781.85	781.85	756.25	756.25	756.25	756.25
N28	5	5	783.20	795.81	757.60	757.60	757.60	757.60
N28	5	6	796.70	809.31	771.10	771.10	771.10	771.10
PL49	4	7	1272.84	1361.90	1197.89	1361.90	1197.89	1361.90

Table 4
Relative cost difference for TNLLP with respect to the optimal solution [%]

Network	n	k	H4B*	SAN*	SAL*	G1	G2	G3	G4	G1$^+$	G2$^+$	G3$^+$	G4$^+$
N14	4	4	2.28	41.24	0	1.61	1.61	1.61	1.61	0	0	0	0
N14	4	5	2.24	39.63	0	1.59	6.21	1.59	6.21	0	0	0	0
N14	4	6	3.98	25.21	3.55	3.41	12.12	3.41	12.12	2.03	12.12	2.03	12.12
* Results for the best methods published in [1].													

Results for TNLLP are given in Tables 2–4 together with results for specialised heuristic H4B, simulated annealing (SAN) and simulated allocation (SAL) from [1]. Results for G1–G4 were obtained after 100 iterations of the algorithm, except PL49, for which results were obtained after 50 iterations. Methods G1$^+$–G4$^+$ were run for 20 iterations.
Table 5 presents convergence of methods. Running times are shown in Tables 6 and 7. We can notice that methods that employ the node cost in the construction phase, i.e. search the shortest path using the installation cost of nodes, the installation cost of edges and the flow cost, give worse results then, one might think less reasonable, method of constructing a solution using just the installation cost of edges and the flow cost.

Table 5
Convergence to the optimal solution [%]

Network	n	k	G1, G3			G2, G4			G1$^+$		
			20*	50*	100*	20*	50*	100*	5*	20*	100*
N14	4	4	10.57	6.32	1.61	10.57	6.32	1.61	1.61	0.00	0.00
N14	4	5	10.39	6.21	1.59	10.39	6.21	6.21	1.59	0.00	0.00
N14	4	6	11.10	7.45	3.41	12.12	12.12	12.12	3.41	2.03	2.03
* Number of iterations.											

We can also find that methods using node switching in their local search give the same results as their counterparts, and run for a slightly longer time. Nevertheless, these methods can be much faster for networks that have high fixed cost of nodes, because they remove in one local search iteration a bunch of edges. Certainly, they can miss the better solution then.

Table 6
Running times for TNLLP [s] (H4B, SAN, G1–G3: Sun Sparc Ultra-4; SAL: PC 800 MHz)

Network	n	k	H4B*	SAN*	SAL*	G1	G2	G3
N14	4	4	3.71	326.74	21.36	18.55	18.69	24.18
N14	4	5	3.77	327.14	3.14	18.40	19.20	23.98
N14	4	6	3.59	318.09	64.08	18.55	7.75	24.26
N14	5	4	3.77	317.61	5.15	13.11	13.06	16.44
N14	5	5	3.78	316.90	19.45	13.01	13.07	16.55
N14	5	6	3.77	324.74	7.46	13.33	12.95	16.84
N28	4	4	35.90	1287.99	78.08	245.01	241.91	300.66
N28	4	5	36.61	1272.05	193.14	242.51	258.77	300.99
N28	4	6	38.65	1271.74	58.66	243.09	84.70	299.33
N28	5	4	35.92	1321.07	58.64	131.13	135.81	160.54
N28	5	5	35.82	1314.30	44.83	130.53	150.39	159.83
N28	5	6	35.00	1304.07	89.97	131.78	133.38	160.58
PL49	4	7	135.46	265.19	146.11	452.67	94.69	514.30
* Results for the best methods published in [1].								

Table 7
Running times for TNLLP [s] – continued (G4, G1$^+$–G4$^+$: Sun Sparc Ultra-4)

Network	n	k	G4	G1$^+$	G2$^+$	G3$^+$	G4$^+$
N14	4	4	23.89	143.55	145.70	151.23	152.01
N14	4	5	25.18	143.01	142.10	150.92	149.89
N14	4	6	11.53	146.95	40.63	163.52	42.81
N14	5	4	16.68	23.51	24.09	24.97	25.01
N14	5	5	16.57	23.76	24.05	24.66	25.03
N14	5	6	16.75	24.20	23.79	24.90	24.95
N28	4	4	297.97	4062.85	4047.29	4164.93	4176.44
N28	4	5	305.95	4080.50	4183.02	4150.94	4289.03
N28	4	6	116.61	4154.86	1961.39	4155.54	2005.43
N28	5	4	160.71	1431.43	1427.85	1522.83	1485.44
N28	5	5	162.96	1435.02	1430.90	1474.59	1506.80
N28	5	6	163.65	1438.98	1491.40	1471.11	1499.58
PL49	4	7	114.45	25439.00	562.95	25736.20	573.21

Finally, edge switching, instead of edge removing, is much better in terms of the final result of GRASP, but it takes a lot of time, because usually there are not as many edges chosen for initial solution (thus not as many to check for removing) as edges not included in the initial solution. In edge switching all the edges of a network must be checked (whether to be removed or included), what can change the method running time for an order of magnitude or even more.
G1 and G3 methods in one case give better results than other heuristic methods from [1] (N14, $n = 4$, $k = 6$), but generally results are 3–5% worse. Their running time is usually longer than the running time for H4B and SAL, but shorter than for SAN. G1$^+$ and G3$^+$ methods give results in 2% neighbourhood ($+/-$) of other heuristic methods from [1], but their running times are very long (cf. PL49).

6. Conclusions

In the paper we proposed a GRASP method implementation to solve topological node and link localisation problem. The problem, consisting in optimal locating of links and nodes under demand constraints, is *NP*-hard, and hence heuristic methods are needed. Heuristics become a **must** for larger networks, for which the running time of exact algorithms becomes infinite.
Numerical examples show that some of GRASP implementations drawn here, which use the demand routing through the shortest path and the local search, are quite efficient and supply good suboptimal solutions. Our study shows that during the construction phase (when we construct the initial solution) it is better not to use installation cost of nodes. Using edges switching in the local search phase instead of edge removing can give better results after fewer iterations, but the time of one iteration is large, and the overall running time of modified method can be order of magnitude larger.
Certainly, the proposed method can still be improved, for example by incorporating more advanced procedures to the construction phase to prepare better initial solution. RCL concept of GRASP can be further deployed by intro-

duction of a function of ranking demands. The promising way might be employing simulated allocation instead of the local search.

References

[1] M. Pióro, A. Jüttner, J. Harmatos, Á. Szentesi, P. Gajowniczek, and A. Mysłek, "Topological design of telecommunication networks. Nodes and links localization under demand constraints", in *17th Int. Teletraf. Congr.*, 2001.

[2] M. Minoux, "Network synthesis and optimum network design problems: models, solution methods and application", *Networks*, vol. 19, pp. 313–360, 1989.

[3] H. H. Hoang, "A computational approach to the selection of an optimal network", *Manag. Sci.*, vol. 19, pp. 488–498, 1973.

[4] D. E. Boyce, A. Farhi, and R. Weischedel, "Optimal network problem: a branch and bound algorithm", *Envir. Plan.*, vol. 5, pp. 519–533, 1973.

[5] R. Dionne and M. Florian, "Exact and approximate algorithms for optimal network design", *Networks*, vol. 9, pp. 37–59, 1979.

[6] B. Gendron, T. G. Crainic, and A. Fragnioni, "Multicommodity capacitated network design", in *Telecommunication Network Planning*, B. Sanso and P. Soriano, Eds. Boston: Kluwer, 1996.

[7] A. Balakrishnan, T. L. Magnanti, and R. T. Wong, "A dual-ascent procedure for large-scale uncapacitated network design", *Oper. Res.*, vol. 37, no. 5, pp. 726–740, 1989.

[8] D. S. Johnson, J. K. Lenstra, and A. H. G. Rinnoy Kan, "The complexity of the network design problem", *Networks*, vol. 8, pp. 279–285, 1978.

[9] M. Pióro, A. Mysłek, A. Jüttner, J. Harmatos, and Á. Szentesi, "Topological design of MPLS networks", in *Globecom*, San Antonio, 2001.

[10] T. A. Feo, M. G. C. Resende, and S. H. Smith, "Greedy randomized adaptive search procedure for maximum independent set", *Oper. Res.*, vol. 42, no. 5, pp. 860–887, 1994.

[11] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures", *J. Glob. Opt.*, vol. 6, pp. 109–133, 1995.

[12] M. G. C. Resende, "GRASP bibliography", http://www.research.att.com/ mgcr/doc/graspbib.pdf

[13] Examples of TNLLP networks, http://www.tele.pw.edu.pl/networks/TNLLP/

Andrzej Mysłek was born in Poland in 1973. He received M.Sc. degrees in telecommunications and in management from the Warsaw University of Technology in 1997 and 2000, respectively. He is now preparing his Ph.D. theses, concerning topological design of telecommunications networks, at the Warsaw University of Technology, in the Institute of Telecommunications.

e-mail: amyslek@tele.pw.edu.pl
Institute of Telecommunications
Warsaw University of Technology
Nowowiejska st 15/19
00-665 Warsaw, Poland