Performance Evaluation of the VNT Reconfiguration Algorithm Based on Traffic Prediction

F. Morales[1], P. Festa[2], M. Ruiz[1*], and L. Velasco[1]

(1) Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

(2) Dipartimento di Matematica e Applicazioni "R. Caccioppoli", Università degli Studi di Napoli Federico II (UNINA), Napoli, Italy

**e-mail: mruiz@ac.upc.edu*

ABSTRACT

In a previous work, the Virtual Network Topology Reconfiguration problem based on Traffic Prediction (VENTURE) was proposed as a means of efficiently adapting core virtual network topology (VNT) to the near-future traffic. Although the benefits obtained by the VENTURE algorithm compared to using a purely reactive VNT reconfiguration approach seems to be clear, margin for improvement still remain and alternative solving methods for the VENTURE problem need to be considered. In this paper, the original VENTURE algorithm is compared against two state-of-the-art metaheuristics for combinatorial network optimization. The two metaheuristics are first presented and then adapted for the VENTURE problem. Finally, the performance of the VENTURE algorithm and the two proposed metaheuristics is numerically evaluated using an exact solving method as reference.

1. INTRODUCTION

The VENTURE problem presented and solved in [1] is an application of cognitive in-operation network planning, where data analytics (prediction) is combined with mathematical optimisation to adapt the VNT to the near future. The problem was mathematically formulated and a heuristic algorithm devised to produce good quality solutions in practical times. The proposed heuristic algorithm was evaluated against purely reactive approaches, offering considerable savings regarding the amount of needed transponders to support the VNT.

Despite the outperforming results of the VENTURE algorithm against purely reactive VNT reconfiguration, a more in-depth assessment of the VENTURE algorithm's performance is needed, including an optimality study against exact solutions and a comparison against state-of-the-art heuristics for combinatorial network optimisation problems. In this paper, we first show the application of the Greedy Randomised Adaptive Search Procedure (GRASP) and Iterated Local Search (ILS) metaheuristics to solve the VENTURE problem. These state-of-the-art, neighbourhood-based metaheuristics have been selected because of their proven quality in solving other combinatorial network optimisation problems [2]. Next, a numerical study is performed to assess the quality of the solutions provided by the original approach, by comparing them against those obtained using the two proposed metaheuristics as well as against those obtained by using an exact solving method.

2. BACKGROUND ON THE VENTURE PROBLEM

In this section, we reproduce the key concepts of the VENTURE problem that are needed to understand this work. The interested reader will find further details about the problem, integer linear programming (ILP), basic heuristics, and performance results in [1].

The problem statement of the VENTURE problem is as follows: Given the current VNT as an extended graph G(N,E), being E the set of current and possible virtual links (*vlinks*) between two routers from N, the set P of transponders of capacity B available in the routers, the current and predicted origin-destination (OD) traffic matrices D and OD, find the reconfigured VNT G*(N,E^*) (where $E^* \subseteq E$) and the paths serving the traffic on G^*. The objective function is to minimise the amount of unserved traffic from matrices D and OD (primary objective) and the number of transponders needed to support the VNT (secondary objective). To ensure enough capacity during the whole reconfiguration period, entries in matrix OD are set as the maximum between the current and the predicted OD traffic.

Besides solving the MILP, a basic heuristic algorithm consisting in three phases can be found in [1]. In summary, these phases works as follows: in phases I and II, those OD pairs with enough predicted capacity to fill transponders above a given boundary usage (e.g. 90%) are processed greedily by allocating a single-hop MPLS path over a direct vlink. To introduce inertia in the current VNT, phase I only considers those OD pairs with an existing direct vlink whereas phase II establishes new vlinks. The unserved bitrate after phases I and II is allocated during phase III by means of an iterative procedure. At each iteration, a constructive phase sequentially serves the remaining bitrate allocating a single MPLS path per OD pair minimising the amount of needed transponders. A local search procedure is executed after the constructive phase, consisting of the sequential removal of one vlink and the rerouting of the disrupted MPLS paths. The re-allocation of these paths is computed ensuring objective cost value decrement. The best solution found is eventually returned.

In the next sections, we present GRASP-based and ILS-based heuristics that aims at improving the aforementioned phase III of the previous basic VENTURE heuristic algorithm.

3. GRASP HEURISTIC ALGORITHM

GRASP is a multi-start, neighbourhood-based metaheuristic that runs a *constructive phase* to build an initial solution, followed by a *local search procedure* to reach a local optimum. The best solution is kept and eventually returned. The constructive phase processes an updated list of *candidates* (solution components) according to some *greedy cost function* (GCF). The greedy/randomness of the solution search can be adjusted by a meta-parameter $\alpha \in [0, 1]$. For a comprehensible survey of GRASP meta-heuristic, please refer to [3].

To solve the VENTURE problem using GRASP, we need to provide the definitions of candidates and the GCF. A candidate u for the VENTURE problem is defined as an OD pair with some remaining capacity to be allocated in the VNT. Formally, it is a tuple $u=<o, u_o>$, where o represents an OD pair and u_o the capacity (bitrate) that remains to be allocated in the VNT.

The pseudocode to compute candidates' greedy costs is detailed in Table 1. Let us first denote the current (under construction) solution x as the tuple $<G(N,E), F>$ containing the current VNT with a set of allocated paths F. The algorithm starts by updating the vlink metrics of the extended topology as a function of the capacity to be allocated (lines 1-7). A low metric is set if a vlink has enough capacity to allocate the needed bitrate (lines 3-4), whereas a high metric is used otherwise. In the case that a vlink has not enough capacity to allocate u_o we distinguish between two cases: a high metric is set in case that the vlink capacity can be increased (with the corresponding transponder utilisation) or the link metric is otherwise forbidden (lines 5-7). Once the vlink metrics are updated the shortest path is computed for OD pair o (line 8), and its total metric is eventually returned as the greedy cost (lines 9-11).

Table 1. GRASP Greedy cost function.

```
   INPUT u=<o, u_o>, x=<G(N,E), F)
   OUTPUT cost
1:  for e ∈ E do
2:    b ← unusedCapacity(e)
3:    if b ≥ u_o then
4:      e.metric ← 1
5:    else
6:      if canIncr(e, u_o - b) then e.metric ← |E|
7:      else e.metric ← +∞
8:  f ← SP(G, o)
9:  cost ← 0
10: for e ∈ f do cost ← cost + e.metric
11: return cost
```

Regarding the local search procedure, we propose the one presented for the original VENTURE algorithm in [1].

Since the computational complexity of updating the vlink metrics is $O(|E|)$ and it is lower than that of the selected shortest path algorithm (Dijkstra's), the worst-case time complexity of the previous GCF is that of the shortest path algorithm: $O(|E|+|N|\cdot\log(N))$. That said, the worst-case time complexity needed to recompute the whole candidate list is $O(|OD|\cdot(|E|+|N|\cdot\log(N)))$.

4. ILS HEURISTIC ALGORITHM

Iterated Local Search (ILS) [4] is a neighbourhood-based metaheuristic that runs a single constructive phase and iteratively perturbs the last local optimum found aiming at finding better solutions. A *strength* meta-parameter is used to adjust the intensity of the perturbations and hence the search strategy: strong perturbations induce diversification whereas small ones induce intensification.

In order to adapt ILS to solve the VENTURE problem, we need to provide constructive, perturbation, and local search procedures. For the sake of fairness, we use the same local search procedure than the one used in GRASP. Table 2 shows the combined constructive/perturbation phase procedure based on the original phase III algorithm presented in [1]. In case that a non-empty candidate OD pair set U is received, the constructive phase is executed starting with the extension of the current topology to a full mesh by creating zero-capacity vlinks (lines 1-5). Otherwise, an input empty U set launches the perturbation procedure that aims at adding candidates OD pairs to such empty set (lines 6-13). Specifically, the current solution is perturbed by removing a random subset of capacitated vlinks from the VNT. The Amount of removed vlinks is determined by the value of the strength parameter (i.e., a percentage of current vlinks). For each removed vlink, the paths traversing the vlink are retrieved, released from the VNT, and added

Table 2. Constructive/Perturbation phase for ILS.

```
   INPUT G, U, strength
   OUTPUT x^best
1:  G*(N, E*) ← G(N, E); F ← ∅
2:  if U ≠ ∅ then
3:    for each e = (i, j) ∉ E | i, j ∈ N, i ≠ j do
4:      E* ← E* U {e}
5:      setCapacity(e, 0)
6:  else
7:    E^rem ← selectAtRandom(E, e.b > 0, strength)
8:    for e ∈ E^rem do
9:      F_e ← getPathsTraversing(e)
10:     for f ∈ F_e do
11:       release(f)
12:       U ← U U u = <f.o, f.b>
13:     setCapacity(e, 0)
14: sort(U, u.u_o, DESC)
15: for each u ∈ U | u.u_o ≠ 0 do
16:     updateMetrics(G*, u.u_o)
17:     f ← SP(G*, u.o, u.u_o)
18:     if f.b < u.u_o AND canIncr(f, G*, u.u_o) then
19:       increaseCap(f, E*, u.u_o)
20:       f.b ← u.u_o
21:     F ← F U {f}
22:     allocate(G*, f)
23: return <G*, F>
```

to U. Finally, the capacity of each removed vlink is set back to zero, thus allowing a possible re-utilisation in the future

The rest of the algorithm is shared by both constructive and perturbation procedures. The set U of unserved OD capacities is sorted greedily, by placing first those OD pairs with higher unserved bitrate (line 14). Once the set of OD pairs is sorted, an iterative procedure is run to process the sorted list of OD pairs aiming at allocating their remaining capacity in the VNT (lines 15-25). At each iteration, the vlink metrics are adapted in order to minimise the transponders needed to allocate the remaining bitrate of a given OD pair (line 15). This minimum cost path is obtained by computing the shortest path algorithm (line 18). In case that a path with the requested bitrate does not exist we check whether the traversed vlinks can increase their capacity by using additional transponders (lines 20-22). The path is allocated and added to the set of paths F (lines 23-24).

The computational complexity of this procedure is $O(|N|^2 \cdot (|E| \cdot \log|N| + R0))$, where $R0$ is the worst-case time complexity to find a feasible lightpath to support every direct vlink. Note that this is equivalent to a single iteration of the phase III algorithm from the original VENTURE algorithm [1].

An important aspect of ILS-based VENTURE is its capability to perform a broader exploration of the solution space in contrast with both the original VENTURE algorithm and GRASP-based VENTURE. The explanation to this is the perturbation procedure: the procedure does not only re-allocate those elements $u \in U$ used as input for phase III but in general any $u = <o, u_o>$ affected by the removal of a vlink, thus opening the possibility of re-allocating OD pair capacity pre-fixed during phases I and II of the algorithm.

5. NUMERICAL RESULTS AND CONCLUSIONS

We now present the performance evaluation of 3 heuristic algorithms: the algorithm from [1] (hereafter referred to as *base*), GRASP and ILS. Additionally, an exact method based using the *branch and bound* (B&B) algorithm [5] to solve the MILP is used to compute optimality gap and scalability of heuristics. We assume that an unlimited number of transponders of B=100 Gb/s are available in the optical layer. A set of 12 reference instances representing different traffic scenarios of a 15-node VNT were solved, divided into three categories depending on the bitrate *granularity* of each OD pair with respect to B. A *high* granularity is defined to represent traffic scenarios where bitrate values are below B, a*verage* granularity to represent OD bitrate centered around B, and finally *low* granularity for traffic scenarios with OD bitrate above B. For each type of granularity, 4 types of traffic transitions from matrix D to OD are considered: a small fluctuation of all the entries in D with uniformly (type *a*) and non-uniformly (type *b*) distributed matrix entries, a large increment of some randomly selected entries in D (type *c*) and a large decrement of some randomly selected entries in D (type *d*).

The three heuristics were implemented in C++ whereas for B&B CPLEX v12.5.1 commercial solver was used. For the particular case of GRASP and ILS different values of their meta-parameters were studied. In addition, 100 repetitions of each instance and algorithm configuration were run with different random seeds. The stopping criterion for the heuristics was set to a maximum running time of 1 minute, whereas for CPLEX the maximum running time was set to 4 hours. All solving methods were executed on a computer with an Intel Core i7-4790K processor, 16GB of RAM and Ubuntu 16.04 operative system.

Table 3 shows the objective cost value of the four different solving methods. For GRASP and ILS, we show the results obtained with the best parameter configuration. For B&B, the best linear relaxation objective cost obtained during the execution of B&B is also shown. We can observe that the heuristic algorithms present an optimality gap between 8% and 9%. A larger gap is observed in all methods for those instances with high bitrate granularity. As a result of the higher bitrate granularity, more routing combinations to serve traffic matrices at a low cost are available, thus making difficult the task to find the best one. Although for high granularity instances B&B outperforms the heuristics in terms of solution quality, these latters are capable of finding quality solutions in only one minute. For average and low granularity instances at least one heuristic approach outperforms B&B both regarding optimality and running time, being ILS the best performing heuristic.

In light of these results, let us focus only on ILS-based VENTURE for the last part of this study, as it appears to be the best solving method according to Table 3. Fig. 1 shows the optimality gap variation when ILS is run vs. base VENTURE. ILS is able to decrease the gap obtained with base VENTURE for any traffic granularity, with the exception of type *b* instances where base VENTURE finds better solutions. In those instances where ILS finds better solutions we obtain an optimality gap reduction between 5% and 20%. Regarding the best parameterisation of ILS, it is worth observing that the optimal strength parameter value seems to present dependency on the OD bitrate granularity.

For low granularity instances, it is better to apply aggressive perturbations (60%-70% of vlinks removed), for average granularity instances the optimal strength is decreased (50%-60% of vlinks removed) reaching the most conservative perturbation strength for high granularity instances (10%-30% of vlinks removed).

In light of these results, we can conclude that ILS is the best heuristic for solving the VENTURE problem, outperforming those heuristics presented in [1]. Note that as a result of the quality of the base VENTURE algorithm, a 20% decrement in terms of optimality gap translates in only 2% decrement of objective cost value (used transponders).

Table 3. Performance of different solving methods.

Instance		Objective function cost					Relative optimality gap			
Granul.	Type	B&B (4 hours)		Heuristics (1 minute)			B&B	Base	GRASP	ILS
		Best bound	Best integer	Base	GRASP	ILS				
High	*a*	142.00	165	167.37	164.91	165.63	16.20%	17.87%	**16.13%**	16.64%
	b	190.75	212	212.08	214.40	213.25	**11.14%**	11.18%	12.40%	11.80%
	c	166.69	187	187.56	186.50	187.42	12.18%	12.52%	**11.88%**	12.44%
	d	136.10	148	157.98	153.81	154.28	**8.74%**	16.08%	13.01%	13.36%
Average	*a*	230.74	250	250.81	255.46	249.23	8.35%	8.70%	10.71%	**8.01%**
	b	357.80	377	376.92	379.77	377.70	5.37%	**5.34%**	6.14%	5.56%
	c	291.07	315	314.57	320.24	313.55	8.22%	8.07%	10.02%	**7.72%**
	d	217.08	235	235.53	237.83	233.53	8.26%	8.50%	9.56%	**7.58%**
Low	*a*	455.44	484	484.14	488.25	480.99	6.27%	6.30%	7.20%	**5.61%**
	b	721.24	745	746.06	749.59	745.81	**3.29%**	3.44%	3.93%	3.41%
	c	590.00	618	618.98	624.15	616.65	4.75%	4.91%	5.79%	**4.52%**
	d	426.22	457	457.26	460.89	453.56	7.22%	7.28%	8.13%	**6.41%**

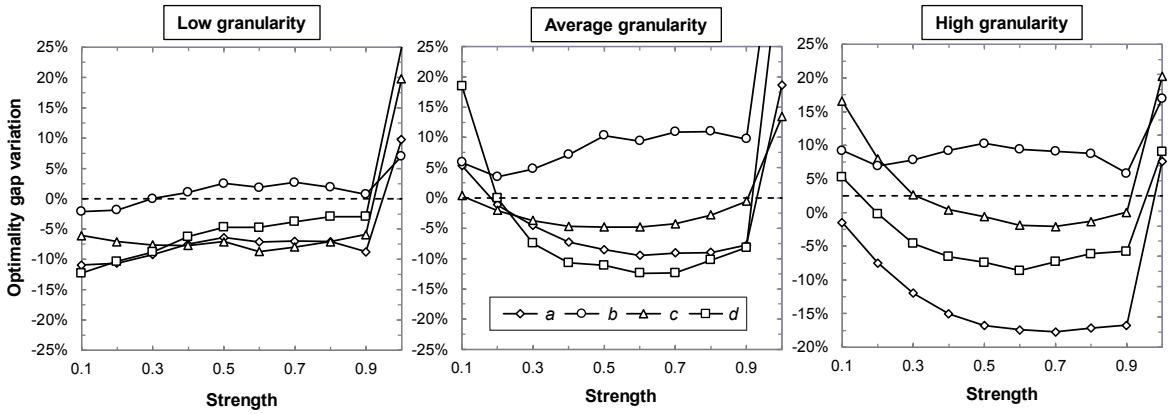

Figure 1. Gap reduction of ILS vs. base VENTURE.

ACKNOWLEDGEMENTS

This work was partially supported by the EC through the METRO-HAUL project (G.A. nº 761727), from the AEI/FEDER TWINS project (TEC2017-90097-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

[1] F. Morales *et al*., "Virtual network topology adaptability based on data analytics for traffic prediction [Invited]," *IEEE/OSA Journal of Optical Communications and Networking (JOCN)*, vol. 9, pp. A35-A45, 2017.

[2] O. Pedrola *et al*., "A GRASP with path-relinking heuristic for the survivable IP/MPLS-over-WSON multi-layer network optimization problem," *Elsevier Computers and Operations Research*, vol. 40, pp. 3174-3187, 2013.

[3] P. Festa and M.G. Resende, "GRASP: An annotated bibliography," *Essays and Surveys on Metaheuristics*, vol. 6, pp. 325-367, 2002.

[4] H.R. Lourenço, O.C. Martin and T. Stutzle, "Iterated local search," *International Series in Operations Research and Management Science*, pp. 321-54, 2003.

[5] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations Research*, vol. 14, pp. 699-719, 1966.