GRASP and ILS Based Meta-Heuristic Approximation Algorithms for Optimal Virtual Network Synthesis (VNS) in Multi-AS Environment

Yong Xue, Alexander Brodsky and Daniel Menasce
George Mason University
Fairfax, Virginia, USA
{yxue2, brodsky, menasce}@gmu.edu

***Abstract*—Emerging Internet architecture utilizes a pluralistic architectural paradigm in which multiple virtual overlay service networks are built on top of the existing interconnected Autonomous System (multi-AS) networks across the world to satisfy diverse and ever-demanding service requirements from end users as well as emerging network applications. However, optimal provisioning and management of virtual networks in the multi-AS environment is still a challenging and unsolved problem. The problem has been extensively studied within the research community as a so-called multi-domain virtual network embedding (MD-VNE) optimization problem, but the current research results so far have been unsatisfactory and lack practical use. This paper proposes a bottom-up virtual network synthesis (VNS) approach to the MD-VNE problem in multi-AS network environment and formulates it into a combinatorial optimization problem. We then propose two trajectory-based meta-heuristic approximation algorithms based on Greedy Randomized Adaptive Search Procedure (GRASP) and Iterative Local Search (ILS) methods together with a simple Greedy Heuristic (GH)-based algorithm as approximate solutions to the multi-AS VNS optimization problem. Experiments with these three approximation algorithms were conducted, and their performances were analyzed quantitatively based on well-defined metrics. The analysis results show the feasibility of the developed approximation solutions for practical solutions to the MD-VNE problem.**

***Index Terms*—MD-VNE, multi-AS Virtual Network Synthesis (VNS), meta-heuristic algorithm, greedy heuristic algorithm, GRASP, ILS, optimization approximation.**

I. Introduction

Emerging Internet architectures utilize a pluralistic architectural paradigm in which multiple virtual overlay service networks are built on top of the existing multi-AS Internet infrastructures to satisfy diverse and ever-demanding services requirements from end users as well as emerging networking technology applications [1]. There are several emerging software defined (SD) networking technologies and applications such as SD-WAN, Network as a Service (NaaS) and overlay QoS service networks [17] [18] [16], in which a virtual network request (VNR) specified by a set of topological and QoS parameters is provisioned across multiple shared substrate networks from multiple independent infrastructure network providers (InP). The virtual network (VN) is instantiated using virtualized node and link resources with certain QoS guarantees from the underlay Infrastructure Network Providers (InP) of terrestrial, 5G mobile, and high-speed SATCOM networks. However, optimal provisioning and management of virtual networks in the multi-AS environment is still a challenging and unsolved problem due to lack of resource exposure/sharing by each InP, and lack of coordination and control across AS boundary as explained in [19] [4].

Multi-domain Virtual Network Embedding (MD-VNE) refers to an optimization problem in the field of network virtualization, which is concerned with efficiently and optimally mapping or embedding virtual network request (VNR) onto a shared set of physical Internet network infrastructures that combined can satisfy all the specified topological and QoS requirements of the VNR. The MD-VNE problem has been studied within the network virtualization research community and has received increasing attention in the past decades [7] [9] [15]. However, all existing MD-VNE solutions are top-down and are faced with challenges in two fronts: 1) how to decompose a VNR into smaller VN requests to map them to different underlay network domains and combine the mapped solutions afterwards; 2) how to gain full topology and resource knowledge of all network domains in a easy and salable way.

Our previous research has introduced a bottom-up virtual network synthesis (VNS) approach for VNE in the multi-AS environment (called mAS-VNS) such that the complexity of the current MD-VNE problem can be reduced to a simplified single domain VNE (SD-VNE) problem logically. Under this paradigm, all network segment resources from multiple InPs are collapsed and aggregated into a single resource pool [19].

There have been two schools of approaches to the MD-VNE problem: 1) finding exact solution via formal optimization formulation solvable by available optimization problem solvers, and 2) obtaining optimal or near-optimal solution through development of proper heuristic or meta-heuristic approximation algorithms [2] [3] [6]. However, formal optimization approach poses challenges in practical use for real-time dynamic network optimization applications due to its high computational costs and the possibility to become an intractable NP-hard problem. For these reasons, many research efforts have turned to heuristic or meta-heuristic-based approximation algorithms recently [3]. To the best of our knowledge, other than various ad-hoc heuristic based algorithms, almost all meta-heuristic algorithms developed for MD-VNE are population-based to include ACO, PSO and GA [11] [5] [12]

[13]. There is a lack of study in trajectory-based meta-heuristic algorithm development for the MD-VNE problem, including Greedy Randomized Adaptive Search Procedure (GRASP), and Iterated Local Search (ILS), which seems to be a natural fit for the mAS-VNS problem.

This paper focuses on developing approximation algorithms for the mAS-VNS optimization problem. The key contributions of this paper include: 1) Development of an optimization framework for developing heuristic and metaheuristic-based approximation algorithms for the mAS-VNS problem, 2) Use of the framework to develop three approximation algorithms including a simple greedy heuristic (GH), a GRASP meta-heuristic and an ILS meta-heuristic algorithms all for the mAS-VNS optimization problem, and 3) Implementation, experimentation and comparative performance assessment of the developed approximation algorithms, including a demonstration of the relative performance of the approximate algorithms compared to the implemented optimal exact solution in [19].

The rest of the paper is organized as follows. Section II provides a brief review of some researches in MD-VNE and related heuristic and meta-heuristic based optimization approximation algorithms. After that, the paper describes a framework for Multi-AS VNS approximation solution development in Section III. Then Section IV describes a simple greedy heuristic algorithm and two meta-heuristic based approximation algorithms to the VNS optimization method. Finally, in Section V, we describe the experiment and performance evaluation results of three approximation algorithms. Section VI provides a brief summary of the research results.

II. Related Work

VN Embedding (VNE) problems have been extensively studied in the network virtualization research community and most solutions are based on either formal optimization or some heuristic methods as summarized in several survey papers [3] [2] [6]. An early research on the MD-VNE problem is presented in [7] [15] and the MD-VNE research has been receiving more attention in the past decades [8] [9]. Some additional examples of the MD-VNE research include the policy-based framework for multi-domain VNE [15], and multi-domain connection stitching techniques [9]. Note that all the results published so far suffer the challenges identified in Section I. Also note that meta-heuristic algorithms seen so far are all population-based including Ant Colony Optimization (ACO) approach [5], Particle Swarm Optimization (PSO) approach [12], Genetic Algorithm (GA) approach [13] plus some variations [11]. In additon, due to the limitation of resource sharing in multi-domain environment, almost all the research proposals use a top-down VNR decomposition approach and are explored under different restrictive assumptions. The resulting solutions have all fallen short to become a complete and practical solution to the MD-VNE problem in real-world [4] [19].

III. Multi-AS VNS Optimization Problem

This section describes a methodology by which some efficient heuristic/meta-heuristic algorithms can be developed as approximate solutions to the mAS-VNS optimization problem. The basic framework is: 1) use SP/VP/InP network mode, but a bottom-up virtual network synthesis approach to avoid top-down network embedding across multiple InP networks; 2) further simplify complexity by collapsing all the InP networks into a single logical InP substrate network (i.e., InP network segment resource pool); and 3) view the mAS-VNS problem as a combinatorial optimization search problem in the search space defined by all possible mappings between the VNR links and potential paths instantiated utilizing network segments provided by the InP substrate networks. Fig. 1 below illustrates the concept to reduce the VNP access network (gateways) and all the InP links (segments) to a single logical InP substrate network shown by the dotted line and exemplary VNR-InP path instantiation as shown by the color-coded VNR link mapping examples across one or more InP networks.

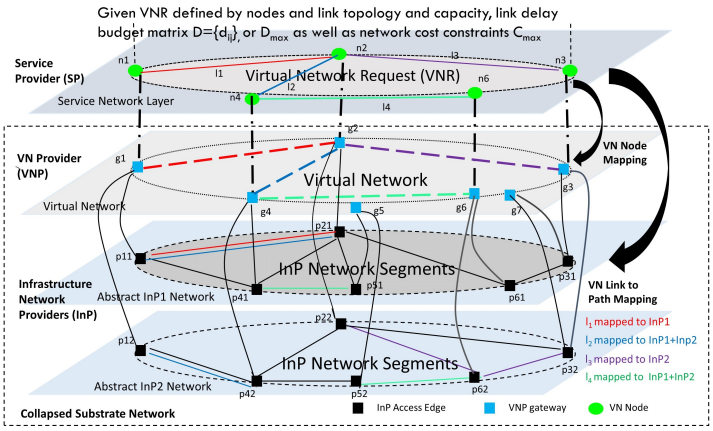

Fig. 1. Multi-AS VNS Solution Framework

To facilitate mAS-VNS solution development, we model a network G using the Attributed Relational Multi-Graph (ARMG) model defined by a 5-tuple

$$G = (N, L, F_L, A_N, A_L)$$

where
N is a finite set of network nodes,
L is a finite set of links connecting pair of nodes in N.
$F_L : L \rightarrow \{\{n_1, n_2\} | \, n_1, n_2 \in N\}$ is a mapping function that maps the links in L to a pair of connecting nodes in N.
$A_N = \{NCap, Loc\}$ is a finite set of attribute functions for node, specifically node capability and location.
$A_L = \{LCap, D\}$ is a finite set of attribute functions for link, specifically link capacity and delay.

In addition, we use super-scripted notations G^V and G^S to represent the VN of VNR and the substrate network of collapsed InPs and VNP networks. Let $p_{ij} = (g_i, s_1, g_h, s_2, \cdots g_j)$ be a non-loop path between gateway nodes g_i and g_j in N^S and $P^S(i, j)$ be the set of all non-loop

paths between gateway nodes g_i and g_j in N^S. We also denote $P_{lnk}(p) = \{s_1, s_2, \cdots, s_{n-1}\}$ be all the links (segments) traversed along path p. Our mAS-VNS optimization objective is:

$$\textbf{Minimize}\ _{m_N \in M_N, m_L \in M_L} Cost(G^V)$$
$$= \sum_{n \in N^V} Cost(m_N(n)) + \sum_{l \in L^V} Cost(m_L(l))$$

where $m_N()$ and $m_L()$ are node and link mapping/synthesizing functions respectively. Since node mapping can be separated from link synthesis in finding a solution, our discussion for the VNR mapping cost of G^V will focus on link synthesis only. Note that if a link l maps to a path p, the following important path properties are true:

$$Cost(m_L(l)) = Cost^S(p) = \sum_{s \in P_{lnk}(p)} Cost^S(s)$$

$$LCap^S(p) = min_{s \in P_{lnk}(p)} LCap^S(s)$$

$$D^S(p) = \sum_{s \in P_{lnk}(p)} D^S(s)$$

Note that a partial and full mapping solution for VNR G_V can be represented as

$$S_k = \{l_1 : p_1, l_2 : p_2, \cdots, l_k : p_k\}\ for\ k \leq |L^V|$$

Therefore, mAS-VNS can be paraphrased as a combinatorial optimization problem based on the fact that each VNS solution is a sequence of link-to-path mapping pairs and that each VNR link $l \in L^V$ that is mapped to gateway nodes g_i and g_j can be instantiated by any one of the paths in $P^S(ij)$. The number of all possible VNS mapping solutions grows exponentially and form the solution search space within which the global optimal solution is to be found, and hence a combinatorial optimization problem.

Since a VNR mapping solution can be incrementally constructed through a sequence of link mapping, for which we have the following equations that will be used in the approximation algorithms to be described in next section.

$$Cost(S_k) = Cost(S_{k-1}) + Cost(p_k)$$

$$D(S_k) = max\{D(S_{k-1}), D(p_k)\}$$

IV. Heuristic and Meta-Heuristic Approximation Algorithms for Multi-AS VNS

Heuristic and meta-heuristic methods are among the most popular and powerful methods for approximation algorithm development to the optimization problem for which either an exact solution does not exist or finding the optimal solution is computationally too costly or intractable. A heuristic algorithm is an approximation method that uses a problem-specific heuristic to incrementally build a solution step by step using some heuristic rules that is expected to produce an optimal or near-optimal solution in the end. A meta-heuristic algorithm is a more general algorithm to approximation of optimization problems, which uses some high-level strategies to guide the effective use of specific heuristic algorithms to improve the performance of the approximation algorithm in order to yield a near-optimal solution more effectively. Trajectory-based meta-heuristic algorithms are a class of optimization algorithms that use iterative improvement strategies to search for the best possible solution. These algorithms explore the search space by creating an initial solution as the start of a trajectory and interactively modify a trajectory to improve its quality following the defined meta-heuristic rules.

In this section, we propose three approximation algorithms for the mAS-VNS optimization problem to include: 1) a Greedy Heuristic (GH) algorithm for incremental approximate solution construction; and 2) two trajectory-based meta-heuristic algorithms using Greedy Randomized Adaptive Search Procedure (GRASP) and Iterated Local Search (ILS) paradigms. Note that the proposed algorithm GH-VNS is a single pass algorithm while the proposed GRASP-VNS and ILS-VNS algorithms are multi-start iterative search algorithms for the mAS-VNS optimization problem. Note that both greedy heuristic algorithm and GRASP/ILS based meta-heuristic algorithms are naturally suitable for the of mAS-VNS optimization search problem where a sequence of best posisble link-path mappings need to be found within reasonable time.

Note that one of basic heuristics used in our approximation algorithms is to instantiate each VN link $l \in L^V$ with next lowest-cost feasible path in the substrate InP networks. To speed up the mapping step, we pre-calculate all k-shortest paths (denoted as $P_k^S(ij)$ for a small select k) between two gateway nodes nodes g_i and g_j using a k-shortest path algorithm using augmented Dijkstra shortest path (SP) algorithm modified for our multi-graph model of the logical InP layer substrate networks.

A. Greedy Heuristic mAS-VNS Approximation Algorithm

Our proposed GH-VNS approximate algorithm utilizes two greedy heuristic rules in incremental approximate solution construction: 1) map each VN link to the next lowest possible cost path between two mapped gateway nodes first because it tends to yield lower mapping cost for the link, thus better total cost for the VNR instantiation in the end; 2) map each link in VN in descending capacity order tends to reduce blocking probability across all the VN links. The algorithm 1 below is the pseudo code of the GH-VNS algorithm.

Heuristic algorithmic rules and implementation strategies for the GH-VNS algorithm are as follows.

1) LinkSort() sorts all links in the VNR in descending order based on the link capacity such that larger capacity links will be attempted first thus minimizing the chance of blocking.
2) At each link instantiating step, we will map each VN link to the next available lowest-cost path in $P_k^S(ij)$ between the two mapped gateway nodes for the link, which does not violate QoS constraints, thus yielding lower incremental cost.

Algorithm 1 GH-VNS: Greedy Heuristic Algorithm for mAS-VNS Optimization.

Description:Given a VNR network, GH-VNS finds an approximation solution to the mAS-VNS optimization problem.

GH-VNS(vnr)

```
cList = LinkSort (vnr) based on link capacity attribute
solution = {}
while not (FullSolutiont(solution)) do
  l = FindNextBestLink (cList)
  p = next least-cost path in P_k^S(ij)
  while not (SatisfyConstraints(solution, l, p)) do
    if P_k^S(ij) ≠ {} then
      p = next least-cost path in P_k^S(ij)
    else
      return no-solution
    end if
  end while
  solution = solution ∪ {l : p}
end while
return solution
```

3) SatisfyConstraints(solution, l, p) checks to see if the current solution plus new link mapping $l : p$ satisfy the capacity, end-to-end delay, and total cost constrain of the current partial solution.

B. GRASP-based Heuristic Approximation Algorithm

Greedy Randomized Adaptive Search Procedure (GRASP) is a trajectory-based multi-start iterative meta-heuristic algorithms for optimal combinatorial search problem [14]. The basic idea of GRASP is to use a two-phase process in which an incremental construction phase in which an initial feasible solution is constructed followed by an improvement phase that utilize a LocalSearch() procedure to find the local optima as the current best solution. Such steps are repeated through multiple iterations and the current best solution is updated at the end of each iteration, which becomes the approximate global optimal solution in the end. To maximize the chance of escaping local optima and exploration of the solution space, a procedure called GreedyRandomizedConstruction() is utilized that introduces greediness, randomness and adaptability into the start solution generation process in each iteration. The GRASP-VNS algorithm utilizes both exploration (multi-start) and exploitation (local search) techniques to search for global optimal solution. The pseudo-code of the GRASP-VNS algorithm is described in Algorithms 2 and 4 below.

Algorithmic rules and implementation strategies for the GRASP-VNS approximation algorithm are noted below:

1) GreedyRandomizedConstruction() implements an incremental construction process for generating new trajectory. To maximize exploration and exploitation of the mAS-VNS solution search, following techniques are employed.

Algorithm 2 GRASP-VNS: Meta-Heuristics Algorithm for mAS-VNS Optimization

Description:Given a VNR network, GRASP-VNS tries to find an approximation solution to the mAS-VNS optimization.

GRASP-VNS(vnr, maxIterations)

```
bestSolution = InitializeSolution()
bestCost = EvaluateCost(bestSolution)
for i = 1 to maxIterations do
  cList = BuildCandidateList(vnr, seed)
  sol = GreedyRandomizedConstruction(cList)
  localOptimum = LocalSearch(sol)
  localOptimumCost = EvaluateCost(localOptimum)
  if localOptimumCost < bestCost then
    bestSolution = localOptimum
    bestCost = localOptimumCost
  end if
end for
return bestSolution
```

GreedyRandomizedConstruction (candidateList)

```
solution = {}
for i = 1 to |L^V| do
  l = L^V[i]
  pList = candidateList[i]
  RCL = BuildRestrictCandidateList (pList)
  p = RandomSelectComponent(RCL)
  solution = solution ∪ {l : p}
end for
return solution
```

A) Greediness: For each mapped link l between gateway nodes g_i and g_j the set $P_k^S(i,j)$ of the k-shortest mapping paths are considered first for lower incremental cost at each step.

B) Adaptation: Instead of considering the first path in $P_k^S(i,j)$ for mapping, GRASP-VNS builds a restricted candidate list (RCL) from the candidate list. In doing so, an increasing number of candidates should be considered following each incremental link mapping to increase the window of available candidate paths so as to avoid potential blocking due to lack of substrate link resources. In our GRASP-VNS algorithm, α-shortest paths ($\alpha \leq k$) are considered for mapping at each incremental VN link mapping, where α is a function of the number of links considered so far (i.e. number of iteration). To make RCL adaptive, the length α parameters is defined as $\alpha = \lceil (i+1)/2 \rceil$ for i-th iteration.

C) Randomness: A mapping path from the α-shortest path list is randomly selected to add into the current partial solution vs. the next shortest path to avoid potential blocking patterns.

2) Note that RandomSelectComponent(RCL) selected link mapping should not violate the bandwidth and delay constraints when added to the current partial solution.

C. ILS-based Heuristic Approximation Algorithm

Iterated Local Search (ILS) algorithm is also a trajectory-based multi-start iterative meta-heuristic algorithm that uses perturbation to escape local optima and maximize the exploration of other local optima in the solution search space [10]. At the same time, by utilizing exploitation techniques of LocalSearch() based on VNS-specific heuristics, the ILS-VNS algorithm tries to explore all local optima dynamically through perturbation to reach the global optimal solution within the search space as described in the high-level pseudo code of the Algorithms 3 and 4 below.

Algorithm 3 ILS-VNS Meta-Heuristics Algorithm:

Description: Given a VNR network,ILS-VNS tries to find an approximation solution to the mAS-VNS optimization.

```
ILS-VNS (maxIterations, perturbationFactor)
  bestSolution = GenerateInitSolution()
  bestCost = EvaluateCost(bestSolution)
  for i = 1 to maxIterations do
    perturbedSolution = PerturbSolution(bestSolution,
    perturbationFactor)
    localOptimum = LocalSearch(perturbedSolution)
    localCost = EvaluateCost(localOptimum)
    if localCost ≤ bestCost then
      bestSolution = localOptimum
      bestCost = localCost
    end if
  end for
  return bestSolution
```

Algorithmic rules and implementation strategies for the ILS-VNS algorithm are below:

1) PerturbSolution() is a key functional component of the ILS which returns a different start solution in search space via a perturbation technique in order to escape local optima and maximize the chance to explore other part of the solution search space for global optimal solution. Various heuristics can be employed as specified by the perturbationFactor parameter. For the VNS problem, the perturbation heuristic we used is randomly select 1/3 of links of the current bestSolution and replace the associated mapped path with a randomly selected t-th $(t \leq k)$ shortest path from the pre-calculated k-shortest path set for that link.
2) GenerateInitSolution() generates a start solution and LocalSearch() is the same procedure utilized in GRASP-VNS.

D. Shared Local Search Algorithm

The basic idea of local search procedure is to take an initial solution and explore the neighbors of the current solution by making some modifications to it and examining the quality of the resulting neighbor solution. The aim of local search is to quickly find the local optima within the local search region and some of the popular techniques include hill climbing, TABU search and variable neighborhood search. The LocalSearch() is a shared function for both GRASP-VNS and ILS-VNS meta-heuristic algorithms.

Algorithm 4 Local Search Algorithm:

Description: LocalSearch() is an algorithm that tries to find the local optimal solution within localized search space.

```
LocalSearch(vnr)
  currentSolution = initSolution
  currentCost = EvaluateCost(currentSolution)
  repeat
    neighborSolution = FindBestNeighbor(currentSolution)
    neighborCost = EvaluateCost(neighborSolution)
    if neighborCost ≤ currentCost then
      currentSolution = neighborSolution
      currentCost = neighborCost
    end if
  until termination condition is met
  return currentSolution
```

The algorithmic rules for the LocalSearch() are:

1) FindBestNeighbor() uses a best-improving strategy and returns most promising neighbor solution in the search space that should be better than the current solution if all constraints are satisfied. For our VNS problem, the best neighbor is selected as follows.
2) Loop through steps below for each link l in VNR that maps to the gateway nodes g_i and g_j.
   - If the current l mapping corresponds to m-th shortest path, then choose the neighbor whose l mapping is replaced by (m-1)-th shortest path from $P_k^S(i,j)\,(m \leq k)$ that does not violate the capacity and QoS constraints, then keep it as current mapping, otherwise try next shorter (i.e., (m-2)-th shortest) path until no more shorter path available, in this case mark this mapping as "best" and will not be further considered for neighbor expansion of link l mapping.
   - Update the current solution with the new l mapping if found in step above
3) Termination condition is met when all the selected neighbor solutions defined by the heuristics above have been exhausted.

V. Experiment and Analysis

In this section, we present the results of experiments and performance analysis of the three approximation algorithms we developed in the context of the following questions we are trying to answer:

1) How do three approximation algorithms (a.k.a, GH-VNS, GRASP-VNS and ILS-VNS) perform compared to the exact optimization algorithm in terms of basic performance metrics, namely run time and cost of the VNS algorithms for a given set of VNR run test data?

2) How well do three approximation algorithms perform in terms of decrease in run-time (i.e., speed-up) vs. increase in cost of the solution (i.e., loss in optimality) for each VNR run?

A. Experiment Setting and Data

In this paper, we use the same test-bed we developed for the formal mAS-VNS optimization solution experiments described in our previous research paper in terms of hardware, software and test data. Specifically, 1000 compatible VNR-InP network pairs of data set (i.e., matching at the node level) are generated as the population and we randomly select 50 of them as sample for the experimental evaluation test run. We leverage the same performance data from the results of mAS-VNS ILP optimization algorithm for our exact vs. approximation solution performance comparison. Note that the same set of the test data are used for approximation algorithm evaluation. Readers can refer to [19] for more details.

B. Evaluation Methods and Metrics

The primary objective of the experiment is to evaluate and compare the performance of the mAS-VNS exact optimal solution and approximate solutions in terms of run-time and network cost. TO that end, we define the following two basic metrics plus two derived metrics for our quantitative performance analysis.

1) Run time denoted by $T(alg)$: The time (in seconds) taken to calculate an exact or approximate solutions by the ILP optimization or approximation algorithms respectively.
2) Cost denoted by $C(alg)$: The total cost (unit cost) of the exact IPL optimization solution and approximation solutions.

Additionally, we define two derived metrics for more advanced performance analysis. The following normalized metrics are defined where A_{opt} represents the optimization algorithm and A_{apr} represents approximation algorithm.

1) Approximation Error Rate (AER): Measures how close the cost of the solution generated by approximation algorithm is to the cost of the exact optimal solution.

$$AER(A_{apr}) = \frac{(C(A_{apr}) - C(A_{opt}))}{C(A_{opt})}$$

Since $C(A_{opt})$ is always less than or equal to $C(A_{apr})$, we have $AER(A_{apr}) \geq 0$ and a larger value means a worse performance.

2) Speed-Up Factor (SF): The ratio between the time taken by the optimal algorithm over the time taken by the approximate algorithm.

$$SF(A_{apr}) = \frac{T(A_{opt})}{T(A_{apr})}$$

Since $T(A_{opt})$ is normally greater than or equal to $T(A_{apr})$, we have $SF(A_{apr}) \geq 1$ and a larger value means a better performance. This is speed up of an approximation algorithm in reference to the exact optimal algorithm.

C. Analysis of the Approximation Algorithm Performance

The following line charts in Figures 2-5 visually show the performance of the three approximation algorithms in terms of two basic and two derived metrics as defined in section V-B. Specifically, Fig. 2 shows the run time of the three approximation algorithms compared to the run time of the exact ILP optimization solution. Fig. 3 shows the costs of the mAS-VNS results from the three approximation algorithms compared to the exact solution from the ILP optimization algorithm. Note that the run-time and cost numbers from the exact solution is based on the ILP implementation described in [19]. Measured from a different perspective, Fig. 4 shows how the speedup stacks among different approximation algorithms. Fig. 5 shows how the Approximation Error Ratio (AER) stacks up among the different approximation algorithms.

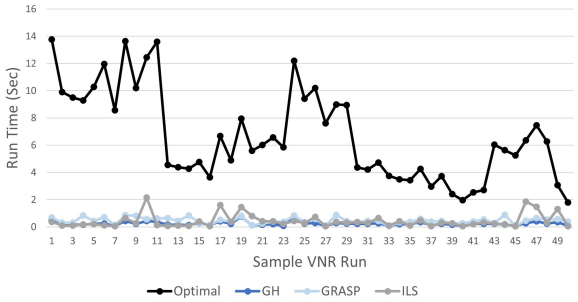

Fig. 2. Run-time for Exact and Approximate Algorithms.

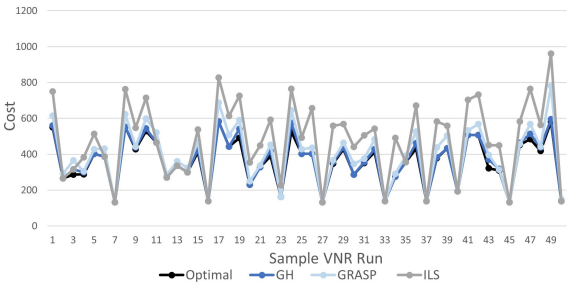

Fig. 3. Costs for Exact and Approximate Algorithms.

From those performance charts in Figures 2-5, the following observations can be made:

1) The run times for all three approximation algorithms are very low and close to each other for all 50 test VNR data runs compared with the exact optimal solution. The ILS-VNS algorithm seems to fluctuate the most, followed by GRASP-VNS with GH-VNS algorithm being the most stable as shown in Fig. 2

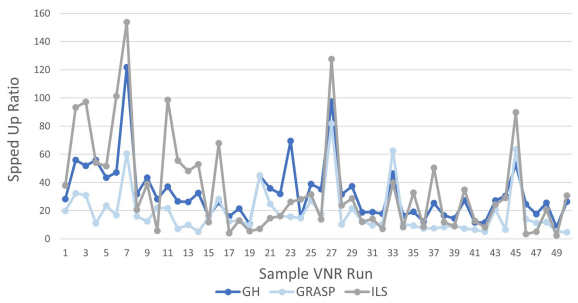

Fig. 4. Speed-up for Approximate Algorithms.

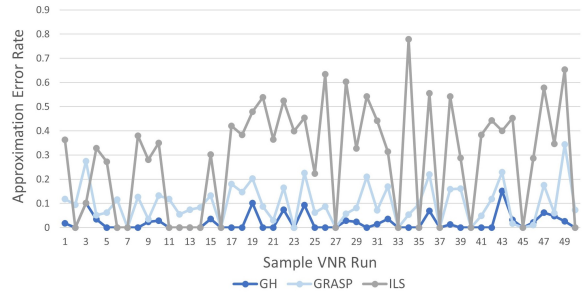

Fig. 5. Error Rate for Approximate Algorithms.

2) The run times for all three approximation algorithms are fractional compared to the exact solution of the ILP optimization algorithm for all test data runs. The runtime of the approximation algorithms is at least one order of magnitude lower as shown in Fig. 4 and Fig. 2.
3) Note that the significant gain in speed for the approximation algorithms is at the cost of relatively small optimality loss for the mAS-VNS solution as shown in Fig. 5
4) Even though for most of the test data runs, the approximate solution's costs are higher than the exact solution from the optimization algorithm, but occasionally the approximation algorithms also generate a result that is optimal as shown in Fig. 3 and Fig. 5.
5) Out of three heuristic/meta-heuristic mAS-VNS algorithms, the GH-VNS algorithms seems to be the most stable and consistent, and performs the best even though the algorithm itself is the simplest compared to the GRASP-VNS and ILS-VNS algorithms.

VI. Conclusion

This paper describes and quantitatively evaluates three heuristic and meta-heuristic approximation algorithms for the mAS-VNS optimization problem as a follow-on study from our previous research on formal modeling and optimization formulation for exact solution to the mAS-VNS problem [19]. The evaluation results are promising and have demonstrated the validity and feasibility of the developed mAS-VNS approximation algorithms to support online and dynamic virtual network provisioning and optimization in multi-AS environment in real world network applications.

References

[1] Anderson, T., Peterson, L., Shenker, S., & Turner, J. (2005). Overcoming the Internet impasse through virtualization. Computer, 38(4), 34-41.
[2] Cao, H., Wu, S., Hu, Y., Liu, Y., & Yang, L. (2019). A survey of embedding algorithm for virtual network embedding. China Communications, 16(12), 1-33.
[3] Cao, H., Hu, H., Qu, Z., & Yang, L. (2018). Heuristic solutions of virtual network embedding: A survey. China Communications, 15(3), 186-219.
[4] Dietrich, D., Rizk, A., & Papadimitriou, P. (2013). Multi-domain virtual network embedding with limited information disclosure. In 2013 IFIP Networking Conference (pp. 1-9). IEEE.
[5] Diallo, M., Quintero, A., & Pierre, S. (2019). An efficient approach based on ant colony optimization and tabu search for a resource embedding across multiple cloud providers. IEEE Transactions on Cloud Computing, 9(3), 896-909.
[6] Fischer, A., Botero, J. F., Beck, M. T., De Meer, H., & Hesselbach, X. (2013). Virtual network embedding: A survey. IEEE Communications Surveys & Tutorials, 15(4), 1888-1906.
[7] Houidi, I., Louati, W., Ameur, W. B., & Zeghlache, D. (2011). Virtual network provisioning across multiple substrate networks. Computer Networks, 55(4), 1011-1023.
[8] Huo, Y., Song, C., Cao, Y., Zheng, J., & Min, J. (2020). A Multi-domain Virtual Network Embedding Approach. In International Conference on Computer Engineering and Networks (pp. 1439-1446). Springer, Singapore.
[9] Li, S., Saidi, M. Y., & Chen, K. (2016). Multi-domain virtual network embedding with coordinated link mapping. In 2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM) (pp. 1-6). IEEE.
[10] Lourenço, H. R., Martin, O. C., & Stützle, T. (2019). Iterated local search: Framework and applications. Handbook of metaheuristics, 129-168.
[11] Madni, S. H. H., Abd Latiff, S. I. M., Coulibaly, Y., & Abdulhamid, S. I. M. (2016). An appraisal of meta-heuristic resource allocation techniques for IaaS cloud.
[12] Ni, Y., Huang, G., Wu, S., Li, C., Zhang, P., & Yao, H. (2019). A PSO based multi-domain virtual network embedding approach. China Communications, 16(4), 105-119.
[13] Pathak, I., & Vidyarthi, D. P. (2017). A model for virtual network embedding across multiple infrastructure providers using genetic algorithm. Science China Information Sciences, 60, 1-12.
[14] Pitsoulis, L. S., & Resende, M. G. (2002). Greedy randomized adaptive search procedures. Handbook of applied optimization, 168-183.
[15] Samuel, F., Chowdhury, M., & Boutaba, R. (2013). Polyvine: policy-based virtual network embedding across multiple domains. Journal of Internet Services and Applications, 4(1), 1-23.
[16] Sitaraman, R. K., Kasbekar, M., Lichtenstein, W., & Jain, M. (2014). Overlay networks: An akamai perspective. Advanced Content Delivery, Streaming, and Cloud Services, 51(4), 305-328.
[17] Song, B., Hassan, M. M., & Huh, E. N. (2012). Delivering IPTV service over a virtual network: a study on virtual network topology. Journal of Communications and Networks, 14(3), 319-335.
[18] Yang, Z., Cui, Y., Li, B., Liu, Y., & Xu, Y. (2019). Software-defined wide area network (SD-WAN): Architecture, advances and opportunities. In 2019 28th International Conference on Computer Communication and Networks (ICCCN) (pp. 1-9). IEEE.
[19] Xue, Y., Brodsky, A., & Menasce, D. (2023). Modeling and Optimization of Virtual Networks in Multi-AS Environment. 12th International Conference on Operation Research and Enterprise Systems(ICORES) 2023.