Explainable optimized solution for the IGP weight design problem

Sebastien Martin
Huawei Technologies Ltd.,
Paris Research Center, France
sebastien.martin@huawei.com

***Abstract*—The Interior Gateway Protocol (IGP) weight design problem is a critical aspect of network optimization, aimed at determining the optimal set of weights assigned to the links in a network to achieve the desired performance objectives. This problem is essential to ensure efficient routing, minimize congestion, and improve overall network reliability. The IGP weight design problem involves complex mathematical modeling and algorithmic approaches to balance multiple factors such as traffic demands, link capacities, and network topology. Effective solutions to this problem can cause confusion among the maintenance team or network managers. One way to tackle this drawback is to find explainable optimization algorithms which aim to enhance the transparency and interpretability of optimization processes. Traditional optimization algorithms often operate as black-box models, providing solutions without insight into their decision-making processes. The goal is to bridge the gap between complex algorithmic operations and human interpretability, thus facilitating better decision-making and trust in automated systems. We propose a way to consider interpretability as an input of the optimization algorithm. The main goal is to compare the optimality loss provided by the interpretability requirement and provide efficient methods to solve the explainable IGP weight design problem. An Integer Linear Program and a local search algorithm are presented and compared with the original problem. An automatic analysis of the solution is proposed to help the maintenance team or network managers.**

***Index Terms*—IGP Weight Problem, Explainability, Mixed-Integer Linear Programming, Local Search**

I. Introduction

The Interior Gateway Protocol (IGP) is a type of protocol used to exchange routing information within an autonomous system (AS). IGPs are designed to handle routing within a single administrative domain, which can be a network managed by a single organization or entity. Common IGPs include the Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and Intermediate System-to-Intermediate System (IS-IS). These protocols help routers within the AS to dynamically discover the best paths for data packets to travel.

The cost of the interior gateway protocol (IGP cost) is a metric used in routing protocols to determine the best route for data to travel within an autonomous system. The cost is typically based on various factors such as bandwidth, delay, and hop count. The lower cost of a link indicates that it is preferred for routing. The default IGP cost value of a link a is $\lfloor \frac{10^8}{c_a} \rfloor$, where c_a is the bandwidth of the link a.

The Interior Gateway Protocol (IGP) weight design optimization problem is a crucial aspect of ensuring efficient and reliable data routing within a network. The primary objective of the IGP weight design optimization problem is to assign appropriate weights to network links to optimize various performance metrics, such as minimizing latency, maximizing throughput, or balancing the load across the network. In the following, the focus is on optimizing the induced link utilization given by the optimized weight. The main goal of this paper is to explore the interpretability and explainability of the IGP weight design optimization problem. Clearly, the proposed approach can be extended to consider many variants of the IGP weight design problem.

The considered IGP weight design ($IGP\text{-}WD$) optimization problem is defined as follows. Given a network and a set of commodities, the goal is to find a weight for each link so that the routing of commodities minimizes the maximum link utilization.

State-of-the-art. The $IGP\text{-}WD$ problem consists of minimizing network congestion and has been extensively explored in the literature for both its *splittable* and *unsplittable* cases. The unsplittable variant consists of ensuring that only one shortest path is induced for each commodity. The splittable variant allows one to split traffic equally between several shortest paths. The Equal-Cost Multiple-Path (ECMP) mechanism is used to split traffic over several paths [10]. Some initial studies addressing shortest-path routing challenges in IP network optimization are [6]. The authors of this paper investigate the problem of designing a survivable VPN-based network using the OSPF routing protocol and propose a compact Mixed-Integer Linear Programming (MILP) model along with several heuristics to tackle the problem. In [8], the authors examine the OSPF weight optimization problem with splittable traffic and a piecewise approximation of the load function. They demonstrate that the problem is NP-hard for a given set of commodities and offer a local search heuristic for its solution. It is also proven to be NP-hard to approximate for both splittable and unsplittable cases [3], [8]. An extension of the local search algorithm is proposed in [9] where the neighborhood is defined to ensure that at least one shortest path changes. In [1], the authors address the unsplittable $IGP\text{-}WD$ problem and analyze the properties of path sets that result in shortest path routing with compatible weights. MILP-based approaches to the problem include that in [4],

[5], [11]. Specifically, compact models are proposed in [11] and [5] for the splittable variant. In [4], Bley introduces the *two-phase algorithm*, an exact method that decomposes the problem into a master subproblem and a client subproblem. The master subproblem generates routing paths, while the client subproblem provides compatible weights, or *conflict inequalities* that prohibit incompatible routing paths if no compatible weights are found. In [2], authors propose other mathematical models to tackle the unsplittable $IGP\text{-}WD$ problem based on Dantzig-Wolfe decomposition. Other works, such as [7], present heuristic methods to solve this problem.

In all of these previous works, several methods are proposed to solve this problem. The major drawback of these methods is how to explain the meaning of the weights generated by the optimization method. For network managers and maintenance teams, it is primordial to understand the meaning of weights to be able to react if unpredictable events, such as failures, loops, and instability, appear on the network.

In the following, we focus only on the unsplitable $IGP\text{-}WD$ problem called $IGP\text{-}WD$ problem where between each pair of vertices, only one shortest path exists.

a) Contributions: We propose a new definition of the problem to consider interpretability constraints as input to the optimization problem. We call this problem the interpretable IGP weight design problem ($interpretable\text{-}IGP\text{-}WD$) problem. We design an integer linear program to model this variant and design a local search algorithm. A comparison between the $interpretable\text{-}IGP\text{-}WD$ and the $IGP\text{-}WD$ solutions and algorithmic performance will be done. We propose a way to analyze the solution produced to help the maintenance team and network managers make efficient use of the network.

II. Problem definition

First, recall the definition of the unsplittable $IGP\text{-}WD$ problem, called $IGP\text{-}WD$ problem in the rest of this paper. A feasible routing implies an unsplittable routing where only one shortest path exists for each commodity.

UNSPLITTABLE IGP WEIGHT DESIGN OPTIMIZATION PROBLEM ($IGP\text{-}WD$ PROBLEM)

Input: A bidirected graph $G = (V, A)$ where each arc $a \in A$ has a *capacity* value $c_a \in \mathbb{N}_+^{|A|}$ and a set K of commodities, where each commodity $k \in K$ is defined as a triplet (s_k, d_k, b_k) representing the source, destination, and bandwidth size of the commodity.

Output: Find a weight assignment $w \in \{1, ..., 2^{16}\}^{|A|}$ that induces feasible routing minimizing the maximum link utilization.

Remark that the weight of each link must be an integer between 1 and 2^{16} due to the link state format to share this weight with other routers [10].

Let us introduce the interpretability concept as input to the optimization problem. For the maintenance team and network manager, the default IGP cost value $\lfloor \frac{10^8}{c_a} \rfloor$ provides a low weight of the link with a high capacity to attract traffic, while a high weight is assigned to the link with a low capacity to avoid high usage, thus inducing a bottleneck in the network. To help interpretability, we consider only some possible weights for each link. Let W_a be the set of possible weights for the link a. For example, this set of possible weights is derived from a set of virtual capacities related to the original capacity. Consider, for instance, a set of potential virtual capacity VC_a for a given link a then the associated set of possible weights is given by $W_a = \{\lfloor \frac{10^8}{vc_a} \rfloor : vc_a \in VC_a\}$.

This set of weights is easy to understand for the maintenance team and the network manager, as each possible weight is associated with a given virtual capacity. Remark that the set of weights can be given in input according to the network size and the type of networks (datacenter, core, aggregation, access, ...). Using a discrete set of weights allows other types of approach to tackle this problem.

The inputs of the interpretable IGP weight design optimization problem are the same as the $IGP\text{-}WD$ problem, where for each link $a \in A$ a set of possible weights W_a is considered.

INTERPRETABLE $IGP\text{-}WD$ PROBLEM ($Interpretable\text{-}IGP\text{-}WD$ PROBLEM)

Input : A bidirected graph $G = (V, A)$ where each arc $a \in A$ has a *capacity* value $c_a \in \mathbb{N}_+^{|A|}$, a set of possible weights W_a, and a set K of commodities where each commodity $k \in K$ is defined as a triplet (s_k, d_k, b_k).

Output : Find a weight assignment $w_a \in W_a$, for all $a \in A$ that induces a feasible routing that minimizes the maximum link utilization.

a) Complexity: In [6], the author proves the NP-hardness of the $IGP\text{-}WD$ problem. For the $interpretable\text{-}IGP\text{-}WD$ problem, in a general case with a large number of possible weights, the problem is equivalent to the $IGP\text{-}WD$ problem since only integer weights are allowed. Remark that if $\max_{a \in A} |W_a|$ is a constant, i.e. not an input value but smaller than a constant, then the interpretable IGP weight design optimization problem can be solved in polynomial time. Enumerate all combinations of weights $|A|^{\max_{a \in A} |W_a|}$ is polynomial, and for a given weight assignment, the quality of the solution can be checked by solving $|K|$ times the shortest path problem.

b) Notations: Let us introduce some notation.

- $\delta^+(v)$, $v \in V$: is the set of outgoing links of the router v.
- $\delta^-(v)$, $v \in V$: is the set of ingoing links of the router v.
- $d(K)$ the set of destinations for all commodities K.

III. Mathematical model

In this section, we present the two integer linear programs for the classical version and the interpretable version. The integer linear program associated with the $IGP\text{-}WD$ problem can be found in several papers [2], [5], [6].

a) Common variables description: Let us introduce the common variables of the two models.

- $x_a^k \in \{0, 1\}$: is equal to 1 if the commodity k is routed through the link a, $\forall a \in A$ and $\forall k \in K$. These variables are called path variables.

- $L \in \mathbb{R}^+$: is equal to the percentage of maximum link utilization associated with the routing policy.
- $u_a^d \in \{0,1\}$: is equal to 1 if all commodities with a destination d are routed through the link a, $\forall a \in A$ and $\forall d \in d(K)$. These variables are called tree variables because they represent a tree for each destination that spans associated sources.
- $r_v^d \in \mathbb{R}^+$: is equal to the distance of the shortest path between v and d, $\forall v \in V$ and $\forall d \in d(K)$.

A. IGP weight design model

Let us introduce the model presented in [2], [5], [6].

a) Variables description: For modeling the classical IGP weight design problem, we must introduce the variable $w_a \in \{1, 2, ..., 2^{16}\}$ for each link $a \in A$ corresponding to the computed weight.

b) $IGP\text{-}WD\text{-}model$: The following model is associated with the $IGP\text{-}WD$ problem.

$$\min L \tag{1}$$

$$\sum_{a \in \delta^+(v)} x_a^k - \sum_{a \in \delta^-(v)} x_a^k = \begin{cases} 1 \text{ if } v = s_k, \\ -1 \text{ if } v = d_k, \\ 0 \text{ otherwise;} \end{cases} \quad \forall v \in V, k \in K, \tag{2}$$

$$\sum_{k \in K} b_k x_a^k \leq c_a L \qquad \forall a \in A, \tag{3}$$

$$\sum_{a \in \delta^+(v)} u_a^d \leq 1 \qquad \forall v \in V, \forall d \in D, \tag{4}$$

$$x_a^k \leq u_a^{d_k} \qquad \forall a \in A, \forall k \in K, \tag{5}$$

$$w_{uv} - r_u^d + r_v^d \geq 1 - u_{uv}^d \qquad \forall uv \in A, \forall d \in d(K), \tag{6}$$

$$w_{uv} - r_u^d + r_v^d \leq M(1 - u_{uv}^d) \qquad \forall uv \in A, \forall d \in d(K), \tag{7}$$

$$x_a^k \in \{0,1\} \qquad \forall k \in K, \forall a \in A, \tag{8}$$

$$r_v^d \geq 0 \qquad \forall v \in V, \forall d \in d(K), \tag{9}$$

$$u_a^d \in \{0,1\} \qquad \forall a \in A, \forall d \in d(K), \tag{10}$$

$$L \geq 0. \tag{11}$$

$$w_a \in \{1, 2, ..., 2^{16}\} \qquad \forall a \in A, \tag{12}$$

where objective (1) minimize the maximum link utilization, the inequalities (2) are the classical constraints of flow conservation. Inequalities (3) represent capacity constraints and make the link to the objective function. Constraints (4) ensure that the union of the paths to reach d is a tree. This constraint is needed to ensure unicity. Inequalities (5) link the x path variables and the u tree variables. Inequalities (6) and (7) ensure that the designed weight induces the right shortest path. Constraints (8)–(11) are domain constraints.

B. Interpretable IGP weight design model

In this section, we provide the model to solve the interpretable version of the IGP weight design problem. This model is inspired by $IGP\text{-}WD\text{-}model$.

a) Variables description: To consider the interpretable version of the problem, we need to introduce one binary variable for each possible weight of each link. Let us introduce a binary variable $\bar{w}_a^i \in \{0,1\}$, for each $a \in A$ and each $w_a^i \in W_a$, equal to 1 if the weight w_a^i is selected for the link a and 0 otherwise.

b) $Interpretable\text{-}IGPWD\text{-}model$: The following model allows solving the $Interpretable\text{-}IGPWD$ problem.

$$\min L \tag{13}$$

$$(2) - (5)$$

$$\sum_{i \in W_{uv}} w_{uv}^i \bar{w}_{uv}^i - r_u^d + r_v^d \geq 1 - u_{uv}^d \qquad \forall uv \in A, \forall d \in D, \tag{14}$$

$$\sum_{i \in W_{uv}} w_{uv}^i \bar{w}_{uv}^i - r_u^d + r_v^d \leq M(1 - u_{uv}^d) \quad \forall uv \in A, \forall d \in D, \tag{15}$$

$$\sum_{i \in W_{uv}} \bar{w}_{uv}^i = 1 \qquad \forall uv \in A, \tag{16}$$

$$\bar{w}_{uv}^i \in \{0,1\} \qquad \forall a \in A, \forall i \in W_a \tag{17}$$

$$(8) - (11)$$

where the objective function (13) has the same meaning as in $IGPWD\text{-}model$. Inequalities (14) and (15) are adaptations of inequalities (6) and (7). Constraints (16) ensure that only one weight is selected for each link.

IV. Local search algorithm

To address large-scale instances, we propose a local search algorithm to solve $IGPWD$-problem and interpretable-$IGPWD$-problem. To be as fair as possible, we consider a similar structure for the two local search algorithms.

The local search algorithm to tackle the $IGPWD$-problem has been tested in the literature. In [8], the authors propose a local search algorithm to solve the IGP weight design model where the neighbors of a solution are defined with a +-1 on the weight of each link. In [9], the authors propose an improvement in which the neighbors of a solution are defined with $+x_a$ or $-y_a$ on the weight of each link $a \in A$ such that $+x_a$ or $-y_a$ are the smallest/greatest value such that at least one shortest path changes. The local search proposed in this section is based on the same neighbor for the $IGPWD$-problem.

In the $Interpretable\text{-}IGPWD$-problem as the weights are given in a discrete set, we consider a neighboring where $+x_a$ and $-y_a$ are rounded to the closest w_a^i, $i \in W_a$, which ensures at least one path change.

The functions associated with the local search algorithms are presented in Algorithm 1. Let us describe each function:

- *GenerateSolution* returns an random weight for the two variants of the problem. For the classical variant, the maximum IGP cost is considered.
- *GenerateNeighborhood* generates solutions in the neighborhood of the current solution. For the interpretability variant, a rounding is needed to consider a valid neighboring. A rounding down if it is below the current weight or a rounding up if it is above the current weight.
- *LocalSearch* is the main function of the algorithm where Local Search run is the maximum number of times to launch the local search with a new starting solution. Local Search iteration represents the number of times that the local search algorithm for a given starting solution tries to improve it.
- *getMLU* evaluates the MLU associated with weights and commodities.

V. Experimental results

In this section, a comparison between the previously proposed methods is described. The goal is to show the MLU loss when the interpretability variant is considered.

All algorithms are implemented in C++ and the mathematical programs are solved using Cplex 12.6.3 with the default settings and a time limit of 1 hour. The experiments are performed on an Intel Core i5-3340M CPU processor of 2.70GHz and 20 GB of memory. For each family of instances, a comparison between five algorithms is done:

- $IGP\text{-}BL$ when the classical IGP cost is used $\lfloor \frac{10^8}{c_a} \rfloor$, for each link $a \in A$.
- WD when $IGP\text{-}WD\text{-}model$ is solved using Cplex.
- $I\text{-}WD$ when $Interpretable\text{-}IGP\text{-}WD\text{-}model$ is solved using Cplex.
- $WD\text{-}LS$ corresponds to the local search algorithm to solve $IGP\text{-}WD$ problem.
- $I\text{-}WD\text{-}LS$ corresponds to the local search algorithm to solve $Interpretable\text{-}IGP\text{-}WD$ problem.

The local search algorithm associated with $WD\text{-}LS$ and $I\text{-}WD\text{-}LS$ is parametrized with Local Search run = 200 and Local Search iteration = 10000. When a local optimum is found, then the algorithm moves to another starting solution.

A. Random instances

Random instances are generated with a given number of routers $\{10, 20, 30, 40, 50, 100\}$, densities $\{0.1, 0.2\}$ and number of commodities $\{10, 20, 30, 40, 50, 100\}$.

a) Graph generator: To ensure connectivity, a random spanning tree is generated and links are added until the target density is not reached. For each link, the capacity is selected in the following set $\{5000, 10000, 50000, 1000000\}$, where the capacities are in MB.

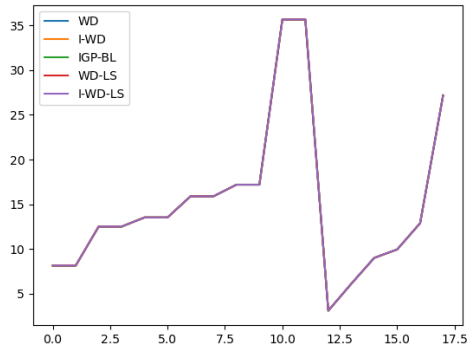
b) Commodity generator: for each commodity, a source and a destination are generated randomly, ensuring that they are two distinct nodes. The bandwidth is computed with the following formulas $1000.0 + 100 \times rand(1, 100)$ where $rand(1, 100)$ provides a random value between 1 and 100.

c) Interpretable IGP weight: To define the interpretable IGP weight set for each link a we consider five virtual capacities $VC_a = \{\frac{c_a}{10}, \frac{c_a}{5}, c_a, 5 \times c_a, 10 \times c_a\}$.

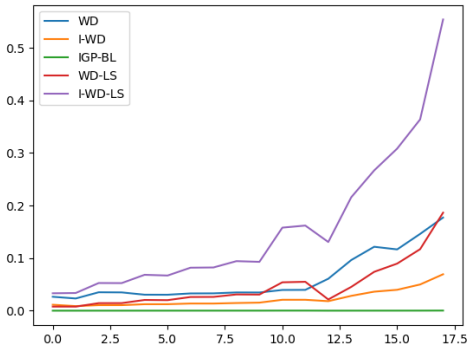
d) Feasibility: First, let us compare the number of instances solved by each method. We have $6 \times 2 \times 6 = 72$ instances.

For random instances, mathematical model can only solve small instances. The interpretable version of the mathematical model is easier to solve than the original one for the solver. This is due to the discrete possible values on weights, which are easier to manage in a Mixed-Integer Linear Program. For the local search algorithm, the interpretable variant is unable to find a feasible solution for one instance. This is probably due to the fact that only discrete weights are available. As we consider the unicity of paths, it can be difficult to find a feasible solution when only a few discrete values of weights are available.

e) Comparison between Integer Linear programming and Local Search algorithms: Now, let us focus on instances where all algorithms find a feasible solution.



(a) Maximum link utilization of solutions



(b) Computational time for each algorithm

Fig. 1: Results on instances solved by all methods

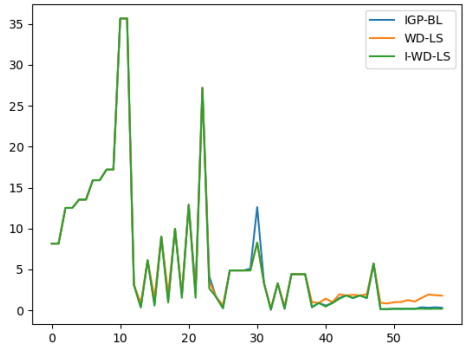
Only small instances (18/72) are solved by all algorithms. In Figure 1 (a), we note that the solution is identical for all of these small instances. Notice that in Figure 1(b), the local search to solve the interpretable version takes important

TABLE I: Number of random instances solved by each algorithm

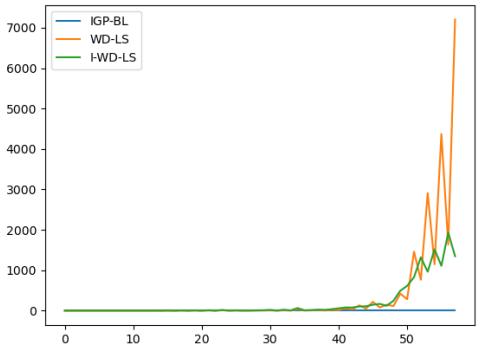
	IGP-BL	*WD*	*I-WD*	*WD-LS*	*I-WD-LS*
Solved instances	59/72	18/72	21/72	72/72	71/72

computational time. This is due to the number of fixed iterations that are too much for small instances. We remark that the computational time associated with the local search for the classical variant grows for the last instances. An interesting remark is that the interpretable variant is easier to solve when integer linear programs are considered. This is probably due to the optimization of the solver on integer variables.

f) Comparison Local Search algorithms: As mathematical programming cannot solve all instances, we now focus on the baseline and the local search algorithms.



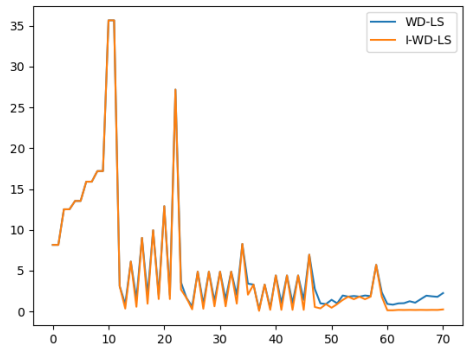
(a) Maximum link utilization of solutions



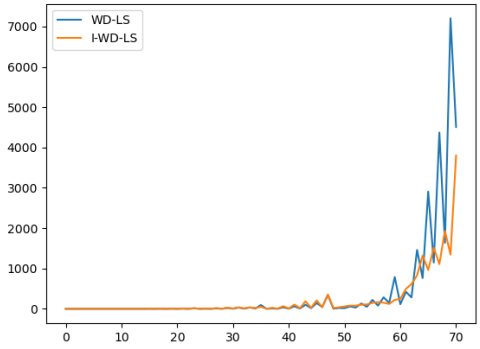
(b) Computational time for each algorithm

Fig. 2: Results on instances solved by the baseline and local search algorithms

In Figure 2, 59 instances are considered. First, we note that the interpretable variant takes less computational time in the largest instances (to the right of Figure 2 (b)) compared to the classical variant. That is the opposite in small instances. If we compare the quality of the solution in Figure 2(a), the baseline obtains a worse solution than the interpretable variant. An unusual finding is that the classical variant finds a solution worth more than the baseline for the largest instances. The discretization of possible weights is more efficient in converging the local search algorithm to a better solution.



(a) Maximum link utilization of solutions



(b) Computational time for each algorithm

Fig. 3: Results on instances solved by local search algorithms

In Figure 3, we show results for the 71 instances solved by the two local search algorithms. Conclusions are similar to the 59 instances presented in Figure 2. The interpretable variant is better for larger instances, either for computational time or quality of the solution.

VI. Understand Relationship between Network and Commodities

To interpret the solution obtained, we can design a virtual network where the virtual capacity vc_a for each link $a \in A$ is set to $\frac{w_a}{10^8}$. For the interpretable version of the problem, the virtual capacity was provided in the input. A comparison between the virtual capacity provided by the previous algorithm and the real capacity provides information on the design of the network. These two values allow us to analyze the trend of link utilization.

- If $vc_a \approx c_a$ then the link a is well used.

- If $vc_a >> c_a$ then the link a is an important link and can become a bottleneck or induce a bottleneck, for example if it fails.
- If $vc_a << c_a$ then this link, in the original setting, is over-provisioned.

The virtual capacities computed by our algorithm are a good indicator of the quality of network design and provisioning according to the considered commodities.

This analysis between the real capacity and virtual capacity helps the network manager expand the capacity or design the network.

Other analyses of the solution can be done to help the maintenance team and network managers understand and trust the solution provided by the proposed method.

VII. Conclusion and perspectives

In this paper, we propose another version of the classical IGP weight design problem. The goal was to provide a new approach that is easy to explain for a network maintenance team. According to our experimental results, we remarked that the loss of quality is limited and sometimes we get a better solution by using the interpretable version of the algorithm. The main reason for the loss of quality comes from the virtual capacities selected in the input.

In future work, the goal will be to introduce humans in the loop to better define the set of virtual capacities and criteria. Another interesting research direction is how to define a good set of interpretable IGP weights that provide a good trade-off between explainability and performance.

References

[1] W. Ben-Ameur, É. Gourdin, "Internet Routing and Related Topology Issues," SIAM J. Discrete Math, 17, 1, pp. 18–49 2003.

[2] A. Benhamiche, M. Chopin, S. Martin, "Unsplittable shortest path routing: Extended model and matheuristic," IEEE 9th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 926–931, 2023.

[3] A. Bley, "Approximability of unsplittable shortest path routing problems," Networks, pp. 23–46, 2009.

[4] A. Bley, "An Integer Programming Algorithm for Routing Optimization in IP Networks," Algorithmica, 60, 1, pp. 21–45, 2011.

[5] A. Bley, B. Fortz, E. Gourdin, K. Holmberg, O. Klopfenstein, M. Pióro, A. Tomaszewski, H. Ümit, "Optimization of OSPF Routing in IP Networks," Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless and Ad Hoc Networks. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 199–240, 1998.

[6] A. Bley, M. Grötschel, R. Wessäly, "Design of broadband virtual private networks: Model and heuristics for the B-WiN," Robust Communication Networks: Interconnection and Survivability, 1998.

[7] C. Dang, C. Bazgan, T. Cazenave, M. Chopin, P.-H. Wuillemin, "Monte Carlo search algorithms for network traffic engineering," Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 486–501, 2021.

[8] B. Fortz, M. Thorup, "Increasing Internet Capacity Using Local Search," Computational Optimization and Applications, pp. 13–48, 2004.

[9] N. Huin, S. Martin, J. Leguay, "Virtual Multi-Topology Routing for QoS Constraints," NOMS 2024 IEEE/IFIP Network Operations and Management Symposium, 2024.

[10] J. Moy, "OSPF Version 2," RFC 1583, 1994.

[11] A. Parmar, S. Ahmed, J. Sokol, "An integer programming approach to the OSPF weight setting problem," Technical Report, School of Industrial & Systems Engineering, Georgia Tech, 2006.

Algorithm 1 Local search algorithm for IGPWD problems

function GENERATESOLUTION($G = (V, A)$, (optional) W)
  if Interpretability **then**
    return $w \leftarrow \{W_a[\mathcal{U}(0, length(W_a))] | a \in A\}$ ▷ random IGP cost assignement
  else
    return $w \leftarrow \{\mathcal{U}(0, 65535) | a \in A\}$ ▷ random values
  end if
end function
function GENERATENEIGHBORHOOD($w, G = (V, A)$, (optional) W)
  $(\delta^+, \delta^-) \leftarrow$COMPUTEDELTAWEIGHTS(G, w) ▷ from [9]
  $\mathcal{W} \leftarrow \emptyset$
  for all $a \in A$ **do**
    $feasible \leftarrow true$
    $w^- \leftarrow w$
    $w_a^- \leftarrow w_a + \delta^-(a)$
    if Interpretability **then**
      if $\{\max w \in W_a : w \leq w_a^-\} \neq \emptyset$ **then**
        $w_a^- \leftarrow \max w \in W_a : w \leq w_a^-$
      else
        $feasible \leftarrow false$
      end if
    end if
    $w^+ \leftarrow w$
    $w_a^+ \leftarrow w_a + \delta^+(a)$
    if Interpretability **then**
      if $\{w_a^+ \leftarrow \min w \in W_a : w \geq w_a^+\} \neq \emptyset$ **then**
        $w_a^+ \leftarrow \min w \in W_a : w \geq w_a^+$
      else
        $feasible \leftarrow false$
      end if
    end if
    if $feasible$ **then**
      $\mathcal{W} \leftarrow \mathcal{W} \cup \{w^+, w^-\}$
    end if
  end for
  return $\mathcal{W}$
end function
function LOCALSEARCH($G = (V, A), K$, (optional) W)
  $MLU \leftarrow \infty$
  $bestWeight$
  for $i \leftarrow 0$ to Local Search run **do**
    $\mathcal{W} \leftarrow$ GENERATESOLUTION(G, W) ▷ generate a new start for the local search to provide a better diversity
    for $i \leftarrow 0$ to Local Search iteration **do**
      $\mathcal{W} \leftarrow$ GENERATENEIGHBORHOOD(w, G, W)
      $w \leftarrow \arg\max_{w \in \mathcal{W}}$ GETMLU(G, w, K) ▷ Compute shortest path of K to get MLU
      $\tilde{MLU} \leftarrow$GETMLU(G, w, K) ▷ Compute shortest path of K to get MLU
      if $\tilde{MLU} < MLU$ **then**
        $bestWeight \leftarrow w$
      end if
    end for
  end for
  return $bestWeight$
end function