Embedding and re-embedding of virtual links in software-defined multi-radio multi-channel multi-hop wireless networks

Lunde Chen, Slim Abdellatif, Armel Francklin Simo Tegueu, Thierry Gayraud

Version of Record: https://www.sciencedirect.com/science/article/pii/S0140366419301690
Manuscript_408885ecfb2ab6d89f450480b0005d68

Embedding and Re-embedding of Virtual Links in Software-Defined Multi-radio Multi-channel Multi-hop Wireless Networks

Lunde Chen[a,b], Slim Abdellatif[a,b,*], Armel Francklin Simo[a,b], Thierry Gayraud[a,c]

[a] *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[b] *Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France*
[c] *Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France*

Abstract

There is rising interest in applying Software Defined Networking (SDN) principles to wireless multi-hop networks, as this paves the way towards bringing the programmability and flexibility that is lacking in today's distributed wireless networks (ad-hoc, mesh or sensor networks) with the promising perspectives of better mitigating issues such as scalability, mobility and interference management and supporting improved controlled QoS services.
This paper investigates this latter aspect and proposes an Integer Linear Programming (ILP) based wireless resource allocation scheme for the provision of point-to-point and point-to-multipoint end-to-end virtual links with bandwidth requirements in software-defined multi-radio multi-channel wireless multi-hop networks. The proposed algorithm considers the peculiarities of wireless communications: the broadcast nature of wireless links which can be leveraged for point-to-multipoint links resource allocations, and, the interference between surrounding wireless links. It also considers switching resource consumption of wireless nodes since, for the time being, the size of SDN forwarding tables remains quite limited. We also consider the case where the requirements of already embedded virtual links evolve over time and propose a re-embedding strategy that meets the new requirements while minimizing service disruption. Genetic Algorithms derived from the ILP formulations are also proposed to address the case of large wireless networks. Our simulation results show that our proposed methods work effectively compared to shortest path based heuristics.

Keywords: virtual link embedding, software-defined networks, wireless multi-hop networks, virtual link re-embedding, point-to-multipoint communication, quality of service

1. Introduction

Applying Software Defined Networking (SDN) design principles to wireless networks can pave the way to the emergence of novel and effective wireless network control applications (routing, network resource allocation, mobility management, energy management, etc.) with diverse expected benefits [1], notably, an improved global network performance, end-to-end network services with enhanced Quality of Service (QoS), etc.

Indeed, under the assumption of an effective topology discovery service [2] that allows SDN controllers to build an updated global and comprehensive view of the network, network control algorithms can leverage on this global and detailed view to derive informed and wise control decisions that are able to accommodate with the dynamicity of the network and flows' QoS requirements. Moreover, the flow level forwarding capability of SDN allows unprecedented fine-grained control on the traffic that is flowing in the network. Some of the prominent works from the literature that attempt to apply SDN to wireless networks in order to dynamically control the traffic for an improved provided QoS are: [3], [4] and [5] respectively in the context of wireless ad-hoc, wireless sensor and wireless mesh networks.

The focus of this work is on the design of resource allocation methods that enable the on-demand provision of network services with QoS requirements in an SDN enabled multi-radio multi-channel multi-hop wireless network. A network service captures the communication needs of an application, and is expressed as a set of end-to-end point-to-point and point-to-multipoint unidirectional virtual (or logical) links (VLs), each with its own bandwidth requirement (the service can be seen as an overlay network required by the application). One important aspect and novelty of this work is that the virtual links can be point-to-multipoint (P2M) and we believe that the most promising use cases are related to the support of P2M virtual links. One possible use case is the support of communications between the decision/command center and a group of firefighters that can be reached via a wireless multi-hop network. Other use cases proposed for 5G hold also [6]. For instance, the distribution of firmware updates to a group of IoT devices, the distribution of traffic and route information to autonomous vehicles located at the same area, and the broadcast of multimedia alerts to population, with some reached via a wireless multihop network.

We first address the embedding of network services, on request arrivals. Then, once embedded, we address the re-embedding of VLs that evolved over time by, either, increasing or decreas-

*Corresponding author

ing their bandwidth requirements, or by having destination nodes joining or leaving point-to-multipoint VLs. In this case, service disruption matters and should be accounted for by the re-embedding algorithm.

Two methods are proposed in this paper for VLs embedding. Both aim at mapping the requested virtual links on the substrate wireless network by computing the data paths that minimize and balance link and switching resource consumption as well as interference between wireless links while satisfying the QoS requirements. They also consider and account for some of the specificity of wireless communications, namely the broadcast nature of wireless links, and the mutual interference caused by transmissions on neighboring links. An Integer Linear Programming (ILP) based formulation method is proposed to compute the optimal allocations for small and moderate size networks as well as an accompanying genetic algorithm for large networks. Two other methods are also proposed for the resource re-embedding of dynamic VLs with service disruption reduction as a key objective. As for embedding, an ILP based algorithm and a genetic algorithm are proposed. A Performance evaluation study is conducted for all methods. The paper is organized as follows. Section 2 reviews previous work from the literature on virtual network resource allocation for wireless networks. Section 3 presents the specifcities of virtual link embedding in wireless multi-hop networks. Then, Section 4 introduces the system model used in our formulations. Section 5 and Section 6 describe the ILP formulations for virtual link embedding and re-embedding, respectively. Section 7 and Section 8 describe the genetic algorithms for virtual link embedding and re-embedding, respectively. Section 9 and Section 10 present the performance analysis of the proposed methods. Finally, Section 11 concludes the paper.

2. Related work

Virtual network embedding has attracted lots of attention in recent years. While most embedding schemes are conceived for wired substrate networks, existing works also considered the case where the substrate network is a wireless one. Table 1 summarizes existing works in the field of virtual link resource allocation, classified according to a set of criteria, among which: the virtual link types (Point-to-Point (P2P) or Point-to-MultiPoint (P2M)), the QoS to meet (bandwidth, delay, packet loss), the network resources considered by the embedding method, the type of methods used for the embedding (ILP based, Genetic algorithm (GA), MILP (mixed integer linear programming), MIQCP (mixed integer quadratic constraint programming)) etc.

As shown in Table 1, it is the combination of the choices made for each feature that makes our proposal unique with respect to existing work. We believe that the effective provision of end-to-end P2MP virtual links with bandwidth requirements in a wireless multihop network is the salient feature of our work. Also, the fact of considering of an SDN based wireless multihop substrate network is new; Besides, it brings its specificities, amongst the importance of accounting for switching resources. Finally, the way we formulate the multicast advantage

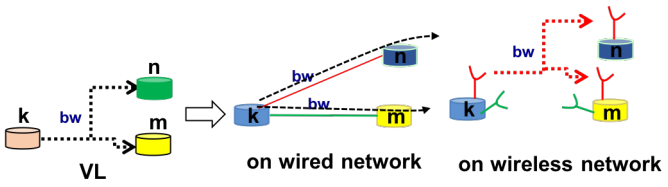

Figure 1: Illustration of the multicast advantage

and interference between wireless links to reduce wireless link resource consumption and to favor spatial reuse frequency is original. Concerning the re-embedding, our work stands out as the only one that take into consideration the dynamicity of requests when applied to wireless substrate networks. For wired substrate networks, there exist some research work as listed in Table 2. Our work stands out, again, in that a wireless SDN substrate network is targeted, with an emphasis on point-to-multipoint communications, as well as switch resources.

3. Some of the specificities of virtual link embedding in wireless multi-hop networks

There has been a lot of work on virtual network embedding for wired network infrastructures. So, one may ask, what makes the problem different in wireless multi-hop networks. In fact, the shared nature of the wireless medium opens some opportunities to save radio resources but also poses some constraints related to interference. For instance, if we consider the case of the P2M VL of Figure 1 crossing Node k and going to Nodes n and m, in a wired context, each transmission related to a virtual link needs to be duplicated. In an equivalent wireless context with nodes equipped with two interfaces, Nodes n and m can be reached with a single transmission on the red channel. An efficient embedding algorithm should whenever possible leverage on the broadcast nature of wireless transmissions to save radio resources. This is what we call from now: the multicast advantage.

On another side, if we consider the wireless network of Figure 2 with multi-radio nodes and different operating channels, there are many alternatives to support the P2M VL: a one-hop data path via the red channel or green channel, or alternatively, a two-hops via the yellow channel then the red one. The impact of these alternatives are different on surrounding links. Clearly, a transmission on the red channel may conflict and prevent much more transmissions from other nodes than a transmission on the green channel. When possible, the transmission on the green channel should be preferred to favor spatial reuse of radio channels. An efficient embedding algorithm should whenever possible mitigate interference on surrounding links to optimize radio resource usage.

4. System model

4.1. Prerequisites on SDN/OpenFlow nodes

We consider an SDN/OplenFlow enabled network. Each node is a wireless OplenFlow switch, which consists of one

Table 1: Classification of virtual link resource allocation schemes for wireless multi-hop networks

	VL type	VL QoS	multi-radio	multi-channel	Network model specificity	Method	Node resources	Support path-split
[7]	P2P	packet loss delay	no	no	link metrics (EATT and EATX)	Incremental virtual network embedding	None	No
[8]	P2P	bandwidth	no	no	mobility	backtracking based heuristic	CPU, storage etc.	Not supported but discussed
[9]	P2P	bandwidth	no	no	interference matrix	heuristic	CPU	No
[10]	P2P	bandwidth	yes	yes	distance based interference model	greedy algorithm and GA	None	No
[11]	P2P	bandwidth	yes	yes	SINR-based interference model	ILP and heuristic	CPU	yes
[12]	P2MP	packet loss	yes	yes	Reliability map	opportunistic rebroadcast	None	No
[13]	P2P	latency	no	no	time-varying link quality and dynamic node workload	deap Q learning	CPU	No
[14]	P2MP	delay	no	no	loops in overlays	ILP and heuristic	Abstract (memory or CPU)	No
[15]	P2MP	delay	no	no	feedback loops in overlay graphs	MILP and MIQCP	transmit power	No
our work	P2P P2MP	bandwidth	yes	yes	conflict graph based interference model	ILP and GA	Flow table entries and group entries	Yes for ILP, No for GA

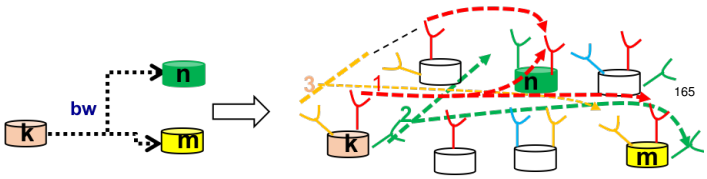

Figure 2: Illustration of the impact of data-paths on the inferred interference

flow table, a meter table and a group table. Each flow table entry is associated to a flow. It combines a match rule that identifies the packets that belong to the flow, the instructions that specify the forwarding actions that apply to the flow of packets. And finally, the counters which maintain the statistics related to the flow. In a meter table, each entry is a traffic rate-policer that can be attached to one or many flow table entries. Finally, a group table entry implements specific methods of forwarding related to broadcasting, load balancing, fail-over, etc. It consists of one or many action buckets and depending on the group type, one bucket is selected (load balancing or splitting), or a copy of the packet is delivered to each bucket (multicast), or a bucket is set to work in back-up (fail-over).

4.2. Network model

Each node in a wireless multi-hop multi-radio multi-channel network is equipped with one or multiple Network Interface Cards (NICs). Each NIC is tuned to a channel and, any two NICs at the same node are tuned to different channels, in order to efficiently and fully make use of radio resources.

We assume that the channel assignment is given and static. There are in total $|\Lambda|$ non-overlapping frequency channels in the system and each node is equipped with q NICs where $q \leq |\Lambda|$. The channel capacity of λ is noted as B_λ.

We model the multi-hop multi-radio multi-channel network as a directed graph $G = (V, E)$ where V is the set of SDN nodes and $E \subseteq V \times V$ the set of bidirectional physical links which operate in half-duplex mode.

To each node $v \in V$, is associated a switching capacity L_v, which is the maximum number of entries (i.e. size limit) of its flow table. The current size of node v flow table is denoted by L'_v. An OpenFlow group table is also considered. A group table entry is either used to duplicate packets belonging to a point-to-multipoint virtual link on different network interfaces or to divide a flow of packets on many interfaces to implement path splitting. Similarly, M_v and M'_v denote respectively the maximum and current size of the group table of node v. We assume that we have already obtained a good channel assignment ζ. ζ assigns to each node $v \in V$ a set of $\zeta(v)$ of $|\Lambda|$ different channels: $\zeta(v) \subseteq \Lambda$.

A pair of NICs can communicate with each other if they are on the same channel and are within the transmission range of each other. In other words, the wireless link $e = ((u, v), \lambda)$, $u, v \in V$ and $\lambda \in |\Lambda|$ belongs to the substrate network, if channel $\lambda \in \zeta(u) \cap \zeta(v)$ and on this latter channel, nodes u and v are within the transmission range of each other. The set of neighbours of v (via any channel) is noted as $N(v)$.

Table 2: Classification of virtual link re-embedding schemes in the literature

Existing Work	VL type	SDN substrate	Evolving Resources	Proactive or Reactive	Solving Method
[16]	P2P	No	Nodes resource, Link bandwidth	Reactive	Heuristic
[17]	P2P	No	Nodes resource (CPU), Link bandwidth	Reactive	ILP
[18]	P2P	No	Node CPU, Link bandwidth	Proactive	Heuristic
[19]	P2P	No	Node CPU, Link bandwidth	Reactive	ILP, Heuristic
Our work	P2P P2MP	Yes	Node switch resources, Link bandwidth	Reactive	ILP, Genetic Algorithm

4.3. Interference model

Our interference model is based on the concept of conflict graphs [21]. A conflict graphs explicitly expresses, for a given frequency band, the presence of pair-wise interference of wireless links (represented as vertices). In other words : simultaneous transmissions on both links either lead to a destructive collision or are prevented by the medium access protocol to avoid the collision. For illustration purposes, Figure 3 shows an SDN-based wireless network substrate composed of six nodes operating on four channels (in different colors) and the associated conflict graphs for the yellow and red channels.

Many options and methods can be adopted to build the conflict graphs, All differ on how they considers two wireless links as interfering. Obviously, two wireless links operating on the same frequency channel that have a node in common interfere (simultaneous transmissions or receptions is not possible). Also, a transmission from a node *s* on a wireless links interferes with receptions on nodes belonging to other wireless links that sit in the interference range of *s*. But, depending on the access technique, things can also be different. For example, with an CSMA/CA based technique for a unicast transmission, an acknowledgement is sent back from the receiving nodes. As a consequence, a unicast transmission on a wireless link form *s* to *d* interferes with receptions at all the nodes sitting in the interference range of node *s* and node *d*. This is not true for multicast transmissions which do not use any acknowledgment. Another salient point of difference is how the interference range of a node is determined by different methods. Some assume that it equals the transmission range and use control packets (hello packets exchange) to establish the list of nodes that sit in the interference range (e.g. [22]). Others consider that the nodes sitting 2 or K hop away from a sending node are subject to interference (e.g. [23]). Some others, such as [24], consider the euclidean distance between wireless links mid-points, etc.

Therefore there are many ways to compute the conflict graphs. All do not lead to the same level of accuracy in capturing interference. But, each can be suitable to a particular network function, since, for example, the accuracy expectation of a channel assignment network function is different from that of a resource allocation network function. Moreover, accuracy comes with a cost in computation time and network control overhead. An approximate near optimal network resource allocation can be suitable in resource constrained wireless networks.
If accuracy is required, as desirable in this work, we believe that this can be achieved by combining information from the knowledge of the network topology (or connectivity graph), information from nodes attributes (transmission power, antenna sensitivity, location if available, etc.) and from wireless links attributes (e.g. link quality assessed by the SINR (Signal to Interference and Noise Ratio), or any other related physical layer metrics or characteristics). The considered SDN based network architecture can clearly help thanks to the centralized view of the network complemented with various nodes and links attributes that the SDN controller sets up by means of its network topology discovery service. Effective techniques that combine different information from different surrounding nodes can be devised to establish accurate conflict graphs. It is however true that with conflict graphs, the cumulative interference of multiple transmissions on distant wireless links is not captured, in contrast to other interference models, such as the SINR interference model which also faces challenges requiring simplifying assumptions to be estimated.

In this paper, to enlarge the applicability of our embedding algorithms, we consider that the conflict graphs are given and provided as an input to our formulation by a dedicated network function at the controller. For the performance analysis of our proposed algorithms, conflict graphs are built following the procedure presented in Section 9.2.

Following the logic of some previous works [25], from the conflict graph, we derive maximal cliques [26]. All pairs of two wireless links in a maximal clique interfere, and hence simultaneous transmissions on these links should be prohibited. So the cumulative bandwidth assignments of the wireless links that compose the maximal clique should be lower than the channel capacity. Also, a wireless link can belong to different maximal cliques in such a situation, the question is how the channel capacity is shared between all the links from the different max cliques. One pessimistic approach is to constrain the cumulative bandwidth assignments of all links to the channel capacity. Such an approach limits the spatial reuse of frequency channels. The optimistic approach assumes that simultaneous transmissions from links belonging to different max cliques are non-interfering links. As a consequence, no additional constraints are required. Other intermediate approaches can be envisioned.

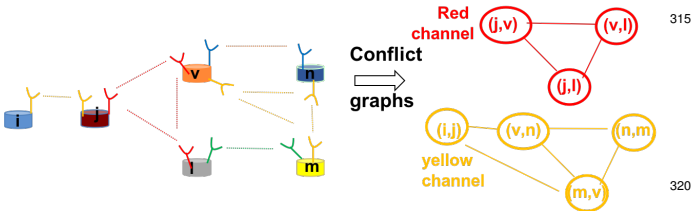

Figure 3: Example of conflict graphs

In our work, we do not stick to any method on how to exploit the maximal cliques to derive interference related constraints. Again, the centralized view of the SDN controller offers a lot of freedom in choosing the level of severity/accuracy in capturing interference between wireless links. For the formulation presented in the paper, without loss of generality, we have chosen the optimistic approach.

Concerning the channel capacity B_λ, the channel capacity was set to a constant value obtained after removing the protocol overhead. Clearly, in a contention based wireless multi-hop network where collisions take place, when increasing the channel load, its effective capacity (goodput) decreases. Since we are considering an online embedding algorithm, at each VLs request arrival, the effective channel capacity could be readjusted.

In the following, we denote the set of maximal cliques as C. We also denote the set of wireless links that form a clique c as E_c, and the nodes of the substrate network in the clique as S_c.

4.4. Virtual Links Request Model

A virtual links request consists of a set of $|K|$ virtual links. Each virtual link $k \in K$ is characterized by:

- a source node $s_k \in V$, and a set of destination nodes $T_k \in V \backslash \{s_k\}$ (when $|T_k| = 1$, the VL is point-to-point, otherwise it is point-to-multipoint);
- a bandwidth requirement of b_k;

The sequence of virtual links requests is noted as $\vec{\mathbf{K}} = [K_1, K_2, ...,]$.

5. ILP formulation for resource embedding

Based on our previous work [28] whose focus was on wired SDN networks, this section describes our ILP formulation of the online virtual links resource allocation on an SDN based wireless multi-hop substrate network. In comparison to the previous work, this formulation adds in many aspects by taking into consideration (1) the broadcast nature of wireless links, which is used as a leverage to efficiently support point-to-multipoint virtual links, as well as, (2) the interferences between surrounding links which is minimized and distributed on different cliques (regions) to improve the admissibility of forthcoming virtual links requests. Below, the variables and problem constraints are listed. Then, the considered objective function is defined.

5.1. Resource-related assignment variables

The resource allocation algorithm provides as output the set of routes with the needed resources to support each of the virtual links (with its required QoS) that compose a request. As mentioned above, two types of network resources are considered : the bandwidth of wireless links and the switching resources (flow table and group table entries). Since VLs may be point-to-multipoint, flow assignment variables and other variables are related to a specific destination of each VL. Our model considers the following variables:

- $f_k^t((v,u),\lambda)$ is an integer variable that represents the bandwidth allocated at link $((v,u),\lambda)$ to the packets of VL k that are flowing from the origin node s_k to a destination node t. More generally, $f_k((v,u),\lambda)$ refers to the amount of bandwidth used on link $((v,u),\lambda)$ by the VL k, whatever the destination. It is set to the maximum of $f_k^t((v,u),\lambda)$ for all $t \in T_k$. Specific to the broadcast nature of wireless medium transmissions, in which one node can deliver a paket to multiple neighbors from one transmission, we also introduce an integer variable denoted as $f_k(v,\lambda)$ that refers to the amount of bandwidth used on channel $\lambda \in \zeta(v)$ by node v to support the VL k. It is set to the maximum of $f_k((v,u),\lambda)$ for all $u \in N(v)$ such as $((v,u),\lambda) \in E$, as is more specifically expressed in Equation 1:

$$\forall k \in K, \forall v \in V, \forall u \in N(v), \forall \lambda \in \zeta(v) \cap \zeta(u) : \\ f_k((v,u),\lambda) \leq f_k(v,\lambda) \quad (1)$$

In this way, multicast advantage is captured in a simple and intuitive manner. This is different from the ILP formulation in [14], in which multicast advantage is captured by loosening the flow conservation restriction by allowing a node to forward at least as much traffic as it has received, which, however, ensures an unfortunate consequence that loops could be created. Additional constraints are needed in order to get rid of such unwanted loops.

- $l_k(v)$ is a binary variable that indicates the number of flow table entries consumed by VL k at node v. An entry is installed in node v flow table if at least one of its adjacent physical links supports the VL. Formally:

$$\forall k \in K, \forall v \in V, \forall u \in N(v), \\ \forall \lambda \in \zeta(v) \cap \zeta(u) : g_k((v,u),\lambda) \leq l_k(v) \quad (2)$$

$$\forall k \in K, \forall v \in V, \forall u \in N(v) : \\ l_k(v) \leq \sum_{\lambda \in \zeta(v) \cap \zeta(u)} (g_k((v,u),\lambda) + g_k((u,v),\lambda)) \quad (3)$$

where $g_k((v,u),\lambda)$ is an intermediate binary variable that equals 1 if some bandwidth is assigned to VL k at link $((v,u),\lambda)$, 0 otherwise. It is derived from some other intermediate variables $g_k^t((v,u),\lambda)$ that, in turn, indicates whether some bandwidth is assigned to the flow of packets of VL k destined to $t \in T_k$ in link $((v,u),\lambda)$ (i.e. $g_k^t((v,u),\lambda) = 0$ if $f_k^t((v,u),\lambda) = 0$ and 1 otherwise).

- similarly, $m_k(v)$ is a binary variable indicating if a group table entry is assigned to VL k at node v. A group table entry is added when splitting a flow of packets belonging to k at node v or when duplicating packets (for point-to-multipoint VLs) on two or more links that operate on distinct channels. This is expressed as:

$$\forall k \in K, \forall v \in V : \quad m_k(v) = \begin{cases} 0 & if \ \sum_{\lambda \in \zeta(v)} g_k(v,\lambda) \leq 1 \\ 1 & otherwise \end{cases} \tag{4}$$

where $g_k(v,\lambda)$ is an intermediate boolean variable that indicates if node v relays packets from VL k on channel λ, whatever its neighbors on this channel. It is derived from the set of previous variables $g_k((v,u),\lambda)$ with $u \in N(v)$ and $\lambda \in \zeta(v) \cap \zeta(u)$, as is more specifically expressed in Equation 5:

$$\forall k \in K, \forall v \in V, \forall u \in N(v), \forall \lambda \in \zeta(v) \cap \zeta(u) : \quad g_k((v,u),\lambda) \leq g_k(v,\lambda) \tag{5}$$

In this way, the embedding process can then be seen as mapping resources onto the complete set of node-channel pair (v,λ), instead of wireless link $((v,u),\lambda)$. Equation 4 could be easily linearized as follows:

$$\forall k \in K, \forall v \in V : \quad 2m_k(v) \leq \sum_{\lambda \in \zeta(v)} g_k(v,\lambda) \leq 1 + |\Lambda| \, m_k(v) \tag{6}$$

- ξ_{max} and ξ_{min} which refer to the maximum and minimum clique utilization after request acceptance (i.e. by taking into account the bandwidth allocations consumed by the virtual links that compose the request).

- l_{max} and l_{min} which similarly refer to the maximum and minimum flow table utilization (when considering all network nodes) after request acceptance.

5.2. Problem constraints

The constraints on bandwidth allocations are described hereafter in Equations 7 to 15. The constraints related to switching resources allocation is given by Inequalities 7 and 8. They simply ensure that the total number of flow and group table entries assigned to VLs composing the request, does not exceed available nodes' flow and group tables entries. Equation 9 reflects the linearization of the *Max* and *Min* operator applied to the variables $l_k(v)$ to get l_{max} and l_{min}.

$$\forall v \in V : \sum_{k \in K} l_k(v) \leq L_v - L'_v \tag{7}$$

$$\forall v \in V : \sum_{k \in K} m_k(v) \leq M_v - M'_v \tag{8}$$

$$\forall v \in V : l_{min} \leq L_v - L'_v + \sum_{k \in K} l_k(v) \leq l_{max} \tag{9}$$

Constraint 10 reflects the linearization of the maximum bandwidth $f_k^t(e)$ allocated to VL k at link $e = ((v,u),\lambda)$, whatever the destination. Equation 11 ensures that the total bandwidth assigned to the substrate wireless nodes that belong to the clique does not exceed the remaining bandwidth of the clique. In this equation, each maximal clique with it's associated channel λ is noted as $(c,\lambda) \in C$, which is composed of E_c, and the residual capacity is $\xi(c) = B_\lambda - \xi'(c)$, with $\xi'(c)$ denoting the bandwidth allocations related to already admitted virtual links on all the physical links that compose the clique c. Equation 12 reflects the linearization of the *Max* and *Min* operator applied to the variables $f_k(v,\lambda)$ to get ξ_{max} and ξ_{min}. Equation 13 presents the usual flow conservation constraints.

$$\forall k \in K, \forall e = ((v,u),\lambda) \in E, \forall t \in T_k : f_k^t(e) \leq f_k(e) \tag{10}$$

$$\forall (c,\lambda) \in C : \sum_{k \in K} \sum_{v \in S(c)} f_k(v,\lambda) \leq B_\lambda - \xi'(c) \tag{11}$$

$$\forall (c,\lambda) \in C : \xi_{min} \leq B_\lambda - \xi'(c) - \sum_{k \in K} \sum_{v \in S(c)} f_k(v,\lambda) \leq \xi_{max} \tag{12}$$

$$\forall k \in K, \forall t \in T_k, \forall v \in V : \quad \sum_{u \in N(v)} \sum_{\substack{\lambda \in \zeta(u) \cap \zeta(v) \\ e_1((v,u),\lambda) \\ e_2((u,v),\lambda)}} (f_k^t(e_1) - f_k^t(e_2)) = \begin{cases} b_k & if \ v = s_k \\ -b_k & if \ v = t \\ 0 & else \end{cases} \tag{13}$$

Equation 14 is a channeling constraint between integer and binary variables: $f_k((v,u),\lambda)$ and $g_k((v,u),\lambda)$. It also constrains the VL k's bandwidth assignment at a physical link to the requested bandwidth b_k. Equation 15 constrains the bandwidth that is assigned to the flow of packets destined to a specific VL's end-point (or destination) within a range of values, in addition to establishing a channeling constraints between binary and integer variables. The inequality on the right side ensures that the bandwidth requirement of the VL is never exceeded. The inequality on the left side directs path-splitting and avoids the multiplication of splits with low bandwidth allocations. Indeed,

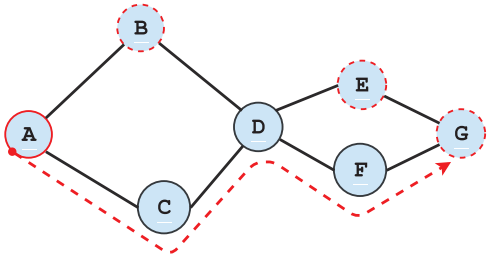

Figure 4: Initial Embedding, with node A as the source, and node G as destination. The evolving of the request requires that node B and E should be added as destinations.

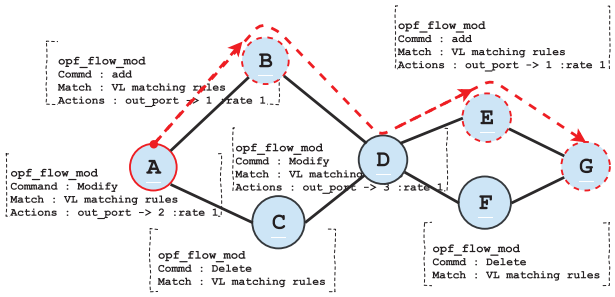

Figure 5: One possible solution: recomputation of a new tree which minimizes resource consumption, but this would introduce service disruptions

if active, path-splitting is feasible only if the bandwidth allocated to the splits respects a minimum threshold b_k^{min}. In practice, b_k^{min} is a ratio of b_k, $b_k^{min} = PS_{ratio} * b_k$ with $PS_{ratio} \in [0, 1]$ (then, $PS_{ratio} \leq 0.5$ when the path-splitting is allowed, and $PS_{ratio} = 1.0$ when it is forbidden).

$$\forall k \in K, \forall v \in V, \forall u \in N(v), \forall \lambda \in \zeta(v) \cap \zeta(u) : \\ g_k((v,u),\lambda) \leq f_k((v,u),\lambda) \leq b_k * g_k((v,u),\lambda) \quad (14)$$

$$\forall k \in K, \forall t \in T_k, \forall v \in V, \forall u \in N(v), \forall \lambda \in \zeta(v) \cap \zeta(u) : \\ b_k^{min} * g_k^t((v,u),\lambda) \leq f_k^t((v,u),\lambda) \leq b_k * g_k^t((v,u),\lambda) \quad (15)$$

5.3. Objective function

Minimize

$$Z_K = \alpha_1 \sum_{k\in K} \sum_{v\in V} \sum_{\lambda\in\zeta(v)} f_k(v,\lambda) + \alpha_2 \sum_{k\in K} \sum_{v\in V} l_k(v) \\ + \alpha_3 \sum_{k\in K} \sum_{v\in V} m_k(v) + \beta_1 \sum_{(c,\lambda)\in C} \sum_{v\in S(c)} \sum_{k\in K} \left|E(c)\right| f_k(v,\lambda) \\ + \beta_2(\xi_{max} - \xi_{min}) + \beta_3(l_{max} - l_{min}) \quad (16)$$

The objective function Z_K of our problem is set to take into account both the resource consumption and the interference introduced by bandwidth allocations on surrounding links. For that, the main objective of our approach is to minimize the total resources required to map virtual links, which is represented by the first three terms that cover respectively links bandwidth, flow tables and group tables resources. In addition, the objective function mitigates the interference between links by avoiding overloading links belonging to cliques with a high number of members. This favors radio resource spatial reuse, increasing the overall available network resources. Finally, the last two terms aim at reducing the disparities of cliques' bandwidth utilization and flow tables' utilization. They also contribute improving flow admissibility. Z_K is then expressed as the weighted sum of those cost components, with α_1, α_2, α_3, β_1, β_2 and β_3 representing their relative significance.

6. ILP formulation for resource re-embedding

6.1. Introduction

VLs request requirements may change over time. New destinations may be added to or removed from a point-to-multipoint virtual link. Also, the bandwidth requirement of a virtual link may change over time. For an already deployed virtual link, a change in its characteristics may require changing a substantial part of its supporting data-paths in order to optimize network resource usage. However, this comes at the cost of a service disruption, the time the re-embedding of the virtual link is accomplished. Obviously, this may be painful, if not acceptable, for some applications.

Let's consider the basic case of the VL depicted in Figure 4, with node A as the source, and node G as the destination. Figure 4 also presents the initial embedding of the VL. If node B and E join the VL as new destinations, one possible approach is to recompute from scratch a new tree that minimizes network resource consumption, as is illustrated in Figure 5. However, this involves six OpenFlow Modification messages to delete the old data-path and install the new one, leading to service disruption that may last some time, especially in a wireless multi-hop network context. Another alternative, illustrated in Figure 6, relies on the initial embedded data-path, and only adds extra data-paths to reach the new destinations. This avoids service disruption and reduces OpenFlow messages exchange but leads to an non optimal network resource consumption. The proposed ILP formulation of the re-embedding technique elaborates on this tradeoff.

6.2. Virtual links request model

The VLs request submitted to the re-embedding method consists of a set of already embedded running VLs from the same or different VLs requests, each with its new characteristics andor requirements. We use k_o and k_e to represent the original and the evolved VLs sub-request respectively, and K_o and K_e to represent the original and the evolved VLs request. Each VL $k_o \in K_o$ has initially been embedded by the resource allocation algorithm and eventually been re-embedded by the algorithm in charge of handling the evolved VLs request.

In our work, there are four basic different scenarios for an evolving VL that can be combined: destination node arrival or departure, bandwidth increase or decrease. Node departures and bandwidth decrease are handled by releasing resources. Next, we focus on the two other scenarios, namely the case of an increase of the VL's bandwidth and the addition of new destination nodes.

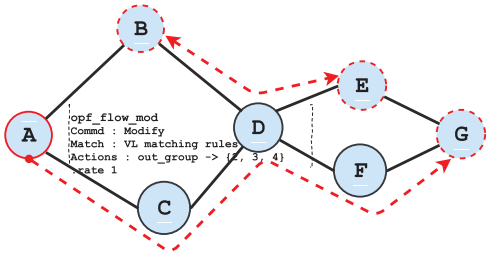

Figure 6: Another possible solution: a solution that is based on the existing embedding tree, which brings minimum reconfiguration overhead and minimizes service disruptions, but consumes more network resources

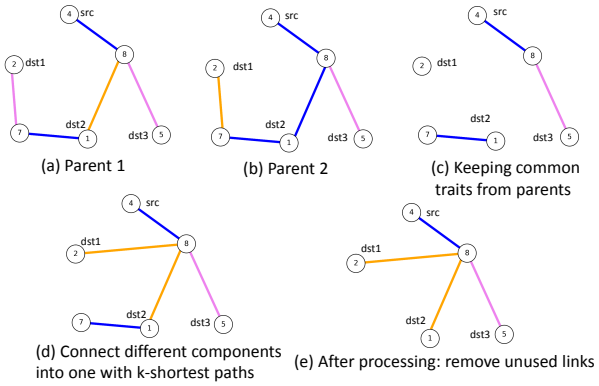

Figure 7: Crossover of two parent trees to get an offspring

6.3. Virtual links re-embedding objective

We use the binary variables $g_k^o(v,\lambda)$ and $g_k^e(v,\lambda)$ to represent original and evolved mapping of a VL k respectively, at each node-channel pair (v,λ). More precisely, $g_k^o(v,\lambda) = 1$ if k is originally embedded onto the node-channel pair (v,λ), otherwise $g_k^o(v,\lambda) = 0$. Likewise, $g_k^e(v,\lambda) = 1$, if VL k is re-embedded onto the node-channel pair (v,λ), otherwise $g_k^e(v,\lambda) = 0$.

The re-embedding cost denoted as $Z^{re}_{K_o,K_e}$ consists of two parts:

- the embedding cost Z_{K_e} which takes into consideration the amount of bandwidth and flow table entries that are consumed, the interference as well as clique load balancing and switch resources balancing, as is defined in Equation 16;
- the reconfiguration overhead (service disruption) which captures service disruption experienced by evolving VLs.

Formally $Z^{re}_{K_o,K_e}$ (that is to minimize) is expressed as follows:

$$Z^{re}_{K_o,K_e} = Z_{K_e} + \rho * \sum_{k\in K_o}\sum_{v\in V}\sum_{\lambda\in\zeta(v)} z^{XOR}_{g_k^o,g_k^e} \tag{17}$$

where

$$\begin{aligned} z^{XOR}_{g_k^o,g_k^e} &= g_k^o(v,\lambda) \oplus g_k^e(v,\lambda) \\ &= g_k^o(v,\lambda) * (1 - g_k^e(v,\lambda)) + (1 - g_k^o(v,\lambda)) * g_k^e(v,\lambda) \end{aligned}$$

Here $z^{XOR}_{g_k^o,g_k^e}$ is the XOR of the original and evolving node channel tuple, which, in fact, corresponds to the presence or absence of service disruption at the scale of one particular node channel tuple. $Z^{re}_{K_o,K_e}$ is hence the linear sum of the consumed resources and the total number of service disruptions. ρ is a constant aimed at scaling the two terms' importance magnitudes. Note that the objective function (that should be minimized in an optimal solution) is linear, as $g_k^o(v,\lambda)$ is not a variable, but a binary constant that is known already when we compute the original embedding results. Hence we are always in an ILP formulation.

Algorithm 1: GA-based Resource Allocation

Input : $G(V,E)$; K; $W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$; $\alpha_1;\alpha_2;\alpha_3;\beta_1;\beta_2;\beta_3$; N_p; N_g; $cxPB$; $mutPB$;
Output: $\chi = [\tau_1, \tau_2, ..., \tau_{|K|}]$ (i.e. the best individual)

```
begin
    P_0 ⟵ InitialPop(G, K, N_p, W)
    P ⟵ P_0
    for ( j_g = 0; j_g < N_g; j_g + + ) {
        for ( j_p = 0; j_p < N_p; j_p + + ) {
            (χ^a, χ^b) ⟵ TournamentSelection(P)
            χ^c ⟵ Crossover(G, K, (χ^a, χ^b), cxPB, W)
            P ⟵ P ∪ Mutation(χ^c, mutPB)
        P ⟵
          PopulationSelection(P, N_p, α_1, α_2, α_3, β_1, β_2, β_3)
    χ ⟵ SelectBestIndividual(P)
```

7. Genetic Algorithm for Embedding

Exact solutions to the considered problem can be obtained by solving our previously presented ILP-based algorithm. However, the complexity of computation, which increases exponentially with the number of parameters (number of nodes, links, radios and channels etc.), might make it practically infeasible for large networks. Therefore, a practically feasible approach is to find a proficient near-optimal solution while sustaining realistic performance. In this section, we present a genetic algorithm based solution to address this aspect. The overall work flow of our GA scheme is described in Algorithm 1. Hereafter we present the detailed algorithm.

7.1. Encoding scheme

To use a GA, it is necessary to choose a representation that defines the genotype of an individual which is conceptually designated as a chromosome. In our case, an individual is a possible solution to a request for resource allocation. Recall that each request K consists of a set of point-to-point and/or point-to-multipoint virtual links. We naturally represent an individual i denoted by χi as a vector of genes where each gene τ_k maps

Algorithm 2: Initial Population Computation

Input : $G(V, E); K; N_p; W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$
Output: $P_0 = \{\chi^i, \forall i \in \{1, ..., N_p\}\ with\ \chi^i = [\tau^i_1, \tau^i_2, ..., \tau^i_{|K|}]\}$

1 **begin**
2 $P_0 \longleftarrow \emptyset$
3 $G'(V', E') \longleftarrow$ `Clone`$(G(V, E))$
4 $W' \longleftarrow$ `Clone`(W)
5 **for** ($i = 0;\ i < N_p;\ i++$) {
6 **foreach** $e \in E'$ **do**
7 $W'[e] \longleftarrow W'[e] \times$ `Random`$(1, 1.5)$
8 **foreach** $k \in K$ **do**
9 $\tau^i_k \longleftarrow$ `ComputeSteinerTree`$(G', W'[e], s_k, T_k)$
10 $\chi^i[k] \longleftarrow \tau^i_k$
11 $P_0 \longleftarrow P_0 \cup \chi^i$

resources assigned to a virtual link $k \in K$. In other words, $\chi i[k] = \tau^i_k$ refers to gene k of individual i. As our GA doesn't take into consideration the case of path splitting, a tree connecting the source s_k to the destination nodes $t \in T_k$, is sufficient to represent a gene. This tree is in fact a subgraph of the substrate network graph G. Each tree is associated with switching resources and links bandwidth respectively allocated at each substrate node and link belonging to this tree.

7.2. Initial population

The first step in the functioning of a GA is the generation of an initial population (Algorithm 1 - Line 2). It is computed by generating a given population size (N_p) with each member of this population encoding an individual representing a possible solution. One important objective is to have a reasonable diversity among the initial population, in order to avoid premature local convergence. In our case, as detailed in Algorithm 2, to generate an individual i (Algorithm 2 - Line 4), we compute the minimum Steiner tree as a routine to build the tree representation of each genes k (Algorithm 2 - Line 9). As constructing Steiner tree is NP-hard, a shortest path heuristic is employed, as is presented in [20]. Note that in the case of multiple links between two nodes, the link with the minimum link cost is sustained for Steiner tree construction. To bring diversity, at each Steiner tree construction, the cost of each link in the substrate network is multiplied by a random factor in the range of $[1, 1.5]$.

7.3. Fitness function

After creating the initial population, each individual is evaluated and assigned a fitness value according to a fitness function. The optimality of a solution is defined by its corresponding fitness value. Equation 18 defines our fitness function.

$$\begin{aligned} F(\chi) =& (F_{bw}(\chi) + F_{openflow}(\chi) + F_{group}(\chi) + F_{interf}(\chi) \\ &+ F_{bw\_balance}(\chi) + F_{sw\_balance}(\chi)) \\ &* \hat{F}_{cliques}(\chi) * \hat{F}_{sw}(\chi) \end{aligned} \tag{18}$$

where

$$F_{bw}(\chi) = \alpha_1 \sum_{\tau \in \chi} \sum_{e \in \tau} \phi(\tau, e) b_\tau$$

$$F_{openflow}(\chi) = \alpha_2 \sum_{\tau \in \chi} \sum_{v \in V} \sigma(\tau, v)$$

$$F_{group}(\chi) = \alpha_3 \sum_{\tau \in \chi} \sum_{v \in V} \eta(\tau, v)$$

$$F_{interf}(\chi) = \beta_1 \sum_{\tau \in \chi} \sum_{(c,\lambda) \in C} \sum_{v \in S(c) \cap S(\tau)} \left|S(c)\right| b_\tau$$

$$F_{bw\_balance}(\chi) = \beta_2 * (max\_bw(C, \chi) - min\_bw(C, \chi))$$

$$F_{sw\_balance}(\chi) = \beta_3 * (max\_sw(V, \chi) - min\_sw(V, \chi))$$

$$\hat{F}_{cliques}(\chi) = 1 + 100 \sum_{(c,\lambda) \in C} \delta(\chi, c)$$

$$\hat{F}_{sw}(\chi) = 1 + 100 \sum_{v \in V} \theta(\chi, v)$$

In the fitness function, $\phi(\tau, e)$, $\sigma(\tau, v)$, $\eta(\tau, v)$, $\delta(\chi, c)$ and $\theta(\chi, v)$ are all indicator functions that take value 0 or 1. Furthermore, the parameters $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2$ and β_3 here are the same as in the objective function of the ILP. $\phi(\tau, e)$ indicates if an edge e in a tree τ supporting a virtual link is transmitting or not. It takes value 1 for all edges in the tree τ except those who use the multicast advantage: in the latter case, $\phi(\tau, e)$ takes value 0. b_τ is the requested bandwidth of a virtual link, and corresponds to the b_k in the ILP formulation. Hence we have $F_{bw}(\chi)$ which is the sum of bandwidth consumed by transmitting links. In the same manner, $\sigma(\tau, v)$ indicates if a node v is included in the tree τ or not, hence consuming one OpenFlow table entry. $\eta(\tau, v)$ indicates if a node v serves as a multicast node or not, hence consuming one group table entry. $F_{interf}(\chi)$ reflects the total interference brought by the instantiated virtual links. For space and clarity reasons, detailed explanations of $F_{bw\_balance}(\chi)$ and $F_{sw\_balance}(\chi)$ are not given here. They correspond to the maximum minus minimum clique bandwidth consumption among all cliques and maximum minus minimum flow table entries consumption among all nodes and can also easily be calculated with $\phi(\tau, e)$ and $\sigma(\tau, v)$.

Those six fitness terms correspond to the objective function in the ILP formulation, i.e. they give the same results when the virtual link embeddings are the same.

Unlike the ILP formulation, the constraints on cliques and switching resources are also included in the fitness function of GA, i.e. the $F_{cliques}(\chi)$ and $F_{sw}(\chi)$ multiplier. In a feasible solution, those two terms should be of value 1. However, in some cases due to the sparsity of feasible solutions, those infeasible solutions should not be removed from the population. In fact, some solutions are more infeasible than others, and should be reflected in our fitness function. To reflect to which extent a solution χ is far from a feasible solution, we penalize those infeasible solutions according to how seriously they violate the clique bandwidth and the switching resource constraints. $\delta(\chi, c)$ indicates if the bandwidth allocations chosen in χ respect clique c bandwidth constraint, i.e. the total bandwidth of transmitting links in c which come from χ, doesn't violate the constraint delimited by the minimum remaining capacity of all links in

c. The 100 here is a large number (compared to 1 as a multiplier) that penalizes violations of constraints. The more we have clique constraint violations for χ, the larger the fitness function $\hat{F}_{cliques}(\chi)$. In the same manner, $\theta(\chi, v)$ indicates if a node respects its switching resource constraint or not, and $\hat{F}_{sw}(\chi)$ reflects to which extent the violation is serious, i.e. the number of nodes that doesn't respect its switching resource constraint. Those two fitness terms correspond to the constraints of switching resources and of cliques in the ILP formulation.

7.4. Selection of parents and crossover scheme

In this stage (Algorithm 1 - Line 6), chromosomes from a population are selected for reproduction (crossover), detailed in Algorithm 3. The operation of selection aims at favoring reproduction and survival of the fittest individuals. We use tournament selection of size 3 to select a pair of chromosomes as the parents to produce an offspring by applying crossover operator between them, with the crossover probability $cxPB$. Its strategy is summarized in Algorithm 3. The idea is simple and consists to pass common traits from parents to offspring according to a specific logic called *Similitude* (Algorithm 3 - Line 4). In order to explain how this primitive works, let $\chi^a = [\tau_1^a, \tau_2^a, ..., \tau_K^a]$ and $\chi^b = [\tau_1^b, \tau_2^b, ..., \tau_K^b]$ be the selected parents. The crossover operator generates a child $\chi^c = [\tau_1^c, \tau_2^c, ..., \tau_K^c]$ by identifying the same links between τ_k^a and τ_k^b for each $k \in K$, and retaining these common links in τ_k^c (as in [29]). According to the definition of the fitness function, the "better" individual has higher probability of being selected as a parent and survive. Thus, the common links between two parents are more likely to represent the "good" traits. However, retaining these common links in τ_k^c may generate some separate sub-trees. Therefore, some other links need to be selected to connect these disconnected sub-trees into a tree. The process is illustrated in Figure 7. First, the same links of τ_k^a and τ_k^b are retained in τ_k^c, in the same way as in [29]. At this moment, τ_k^c could be disconnected and divided into several components. Moreover, to maintain diversity among solutions, instead of connecting separated components each time with the shortest path, we adopt a random k-shortest path (with k $\leq$ 3). Note that in the case of multiple links between two nodes, the link with the minimum link cost is used for k-shortest path construction. This process is repeated until τ_k^c becomes connected (Algorithm 3 - Line 5 to 6). Finally, a post-processing can be required to remove isolated branches of the tree that contain neither the source node nor destination nodes, as shown in Figure 7 (d) and (e).

As the cross-over is carried out for VLs in a one-by-one manner, the function updateProhibitiveLinkCost (Algorithm 3 - Line 10) assigns an infinity link cost to links that belong to cliques with no bandwidth left for future VLs, and to links with one end node that has no flow table entries left. In this way, infeasible solutions are excluded for search when possible, boosting the efficiency of exploration in the solution space.

7.5. Mutation schemes

When a new offspring is produced, the mutation operation is performed according to the mutation probability $mutPB$ (Algorithm 1 - Line 8). We identify two types of mutations that could be very helpful, i.e. (1) mutation based on link breaking and reconnection, (2) mutation based on channel transition. The action of mutation gives more chances of getting rid of local sub-optimal solutions. For the first type of mutation (as shown in Figure 8), the procedure randomly selects a link in the tree and remove it to create two separate sub-trees; then, it re-connects these separate sub-trees using a random k-shortest path. An after-processing could always be needed to cut off unused branches. The second type of mutation come from the importance of channels in our problem. That is, when a selected link to mutate in the tree corresponds to a multi-link in the substrate network, it's possible to directly change the channel of this link, as shown in Figure 9. This could lead us to finding more opportunities of multicast advantage, or a better load balancing among cliques.

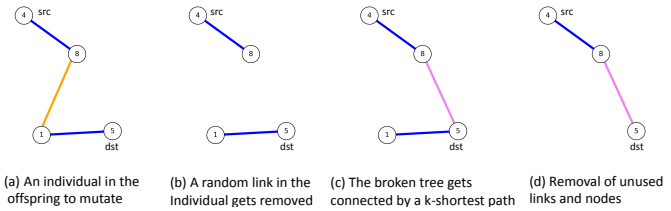

Figure 8: Mutation scheme I: break-down and re-connection

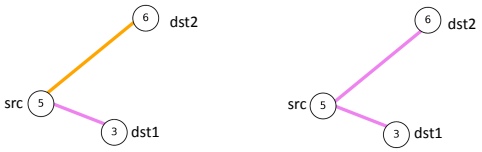

Figure 9: Mutation scheme II: channel mutation

7.6. Balanced resource allocation with dynamic link cost

We set the normal link cost of $e = ((v, u), \lambda)$ as $w_e = \alpha_1 + \alpha_2 + \beta_1 \left|E(c)\right|$. If this static manner of defining the link cost is used across all requests in $\vec{\mathbf{K}} = [K_1, K_2, ...]$, cliques with low link costs are always favored in comparison to cliques with high link costs, regardless of their current load, leading to unbalanced cliques utilizations. Although in our fitness function the clique utilization balancing is taken into consideration, it's inefficient if most candidate solutions in GA lead to an unbalanced situation. To mitigate this, we give a dynamic version of link cost, as shown in Algorithm 4. The idea of the algorithm is that if the clique utilization of a clique c is among the top N_{top} most loaded, the link cost of links that form c should be multiplied by a factor of 1.1. If the clique c is yet being used in the current embedding of K (i.e. $\chi^K \cap c \neq \emptyset$, as shown in Line-11 of Algorithm 4), then an even higher multiplying factor (i.e. 1.5) should be given to links in c, before calculating the embedding solution of K_{next}. In this way, those most used cliques will be unfavored in the embedding of forthcoming requests, due to their high link costs. Note that the increase of link cost can be accumulated over time, i.e. if a clique stays always among the top most loaded from K_i to $K_{i+\Delta}$, its links costs

Algorithm 3: Crossover Scheme

Input : $G(V,E); K; \chi^a; \chi^b; cxPB;$
$W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}];$
Output: $\chi^c = [\tau^c_1, \tau^c_2, ..., \tau^c_{|K|}]$

begin
  $G'(V',E') \longleftarrow$ `Clone`$(G(V,E))$
  $W' \longleftarrow$ `Clone`(W)
  if *(*`Random`$(0,1) < cxPB$*)* **then**
    foreach $k \in K$ **do**
      $\tau^c_k \longleftarrow$ `Similitude`(τ^a_k, τ^b_k)
      while `isNotConnected`(τ^c_k) **do**
        $\tau^c_k \longleftarrow$ `randomKShortestPath`(G', W', τ^c_k)
      $\chi^c[k] \longleftarrow \tau^c_k$
      `updateProhibitiveLinkCost`(G', W', τ^c_k)

will be increased $\Delta + 1$ times, until the clique is removed from the top most loaded list, at which point the normal link cost is given to the links in the clique. Inverse actions are carried out on the N_{top} least used cliques. At each iteration, links in other cliques are given normal link cost.

8. Genetic Algorithm for Re-embedding

GA for re-embedding takes the same steps as GA for embedding, with some differences. Hereafter we present what is different in GA for re-embedding.

8.1. Fitness function for re-embedding

The fitness function for re-embedding is defined as:

$$F^{re}(\chi) = Z^{re}_{K_o,K_e}(\chi) * \hat{F}_{bw}(\chi) * \hat{F}_{sw}(\chi) \tag{19}$$

Following the same logic as in Section 7.3, $Z^{re}_{K_o,K_e}(\chi)$ is calculated by computing the objective function defined in Equation 17 for a chromosome χ. $\hat{F}_{bw}(\chi)$ and $\hat{F}_{sw}(\chi)$ are defined in the same manner as in Equation 18.

8.2. Population initialization for re-embedding

The population initialization for re-embedding is shown in Algorithm 5. Two objectives should be achieved for this task: (1) Diversity among populations, in order to avoid premature local convergence. (2) Similarity to original links χ_o, so that the evolved mapping results keep as many similar traits to the original embedding results as possible in order to minimize service disruptions.

To achieve the first objective, the cost of each link in the substrate network is multiplied by a random factor in the range of $[1, 1.5]$, in the same manner as in GA for embedding. To achieve the second objective, for those links constituting the original VL χ_o, the cost of each of them is multiplied by a small random factor in the range of $\left[\gamma^{min}_o, \gamma^{max}_o\right]$ (for example, in the range of $[0.01, 0.1]$). Afterwards, we compute the minimum Steiner tree as a routine to build the tree representation of each gene k of an individual χ.

Algorithm 4: Balanced Resource Allocation With Dynamic Link Cost

Input : $G(V,E); \vec{\mathbf{K}}; C; N_{top}; \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3,$
$N_p, N_g, cxPB, mutPB$
begin
  foreach $e \in E$ **do**
    $W[e] \longleftarrow \alpha_1 + \alpha_2 + \beta_1 \left|E(c)\right|$
  foreach $K \in \vec{\boldsymbol{K}}$ **do**
    $\chi^K \longleftarrow$ `geneticAlgorithm`$(G, K, W, \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3, N_p, N_g, cxPB, mutPB)$
    $C_{most} \longleftarrow$ `mostUsedCliques`(C, N_{top})
    $C_{least} \longleftarrow$ `leastUsedCliques`(C, N_{top})
    $C_{normal} \longleftarrow C - C_{most} - C_{least}$
    foreach $c \in C_{most}$ **do**
      if $c \cap \chi^K \neq \emptyset$ **then**
        foreach $e \in c$ **do**
          $W[e] \longleftarrow W[e] \times 1.5$
      else
        foreach $e \in c$ **do**
          $W[e] \longleftarrow W[e] \times 1.1$
    foreach $c \in C_{least}$ **do**
      if $c \cap \chi^K = \emptyset$ **then**
        foreach $e \in c$ **do**
          $W[e] \longleftarrow W[e] \div 1.5$
      else
        foreach $e \in c$ **do**
          $W[e] \longleftarrow W[e] \div 1.1$
    foreach $c \in C_{normal}$ **do**
      foreach $e \in c$ **do**
        $W[e] \longleftarrow \alpha_1 + \alpha_2 + \beta_1 \left|E(c)\right|$

8.3. Crossover for re-embedding

The crossover for re-embedding is shown in Algorithm 6. The performed actions include:

- Retaining these common links in the two parents with a high probability. To this end, in the substrate network, the link costs of those common links are multiplied by a random near-to-zero factor in the range of $\left[\gamma^{min}_{comm}, \gamma^{max}_{comm}\right]$ (e.g. around 0.05).
- At the same time, it is always important to guarantee the similarity to original links as well as the diversity of solutions. To this end, the link cost of original links are multiplied by a small random factor in the range of $\left[\gamma^{min}_o, \gamma^{max}_o\right]$ (for example, in the range of $[0.01, 0.1]$).
- Other links are multiplied by a random factor in the range of $[1, 1.5]$.

Afterwards, we compute the minimum Steiner tree on the substrate network with adjusted link costs. Note that here we

Algorithm 5: Initial Population Computation for Re-embedding

Input : $G(V,E); \chi_o; K_e; N_p; \gamma_o^{min}; \gamma_o^{max}; W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}];$
Output: $P_0 = \{\chi^i, \forall i \in \{1, ..., N_p\}\}$, with $\chi^i = [\tau_1^i, \tau_2^i, ..., \tau_{|K|}^i]\}$

begin
  $P_0 \longleftarrow \emptyset$
  $G'(V', E') \longleftarrow$ `Clone`$(G(V,E))$
  $W' \longleftarrow$ `Clone`(W)
  for ($i = 0;\ i < N_p;\ i++$) {
    foreach $k \in K_e$ **do**
      $W' \longleftarrow$ `Clone`(W)
      foreach $e \in \chi_o[k]$ **do**
        $W'[e] \longleftarrow W'[e] \times$ `Random`$(\gamma_o^{min}, \gamma_o^{max})$
      foreach $e \in E' - \chi_o[k]$ **do**
        $W'[e] \longleftarrow W'[e] \times$ `Random`$(1, 1.5)$
      $\tau_k^i \longleftarrow$ `ComputeSteinerTree`$(G', W'[e], s_k, T_k)$
      $\chi^i[k] \longleftarrow \tau_k^i$
    $P_0 \longleftarrow P_0 \cup \chi^i$

Algorithm 6: Crossover Scheme for Re-embedding

Input : $G(V,E); \chi_o; K_e; \chi^a; \chi^b; cxPB; \gamma_{comm}^{min}; \gamma_{comm}^{max}; \gamma_o^{min}; \gamma_o^{max}; \gamma_{normal}^{max}; W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}];$
Output: $\chi^c = [\tau_1^c, \tau_2^c, ..., \tau_{|K|}^c]$

begin
  $G'(V', E') \longleftarrow$ `Clone`$(G(V,E))$
  if *(*`Random`$(0,1) < cxPB$*)* **then**
    foreach $k \in K$ **do**
      $W' \longleftarrow$ `Clone`(W)
      $\tau_k^c \longleftarrow$ `Similitude`(τ_k^a, τ_k^b)
      foreach $e \in \tau_k^c$ **do**
        $W'[e] \longleftarrow W'[e] \times$ `Random`$(\gamma_{comm}^{min}, \gamma_{comm}^{max})$
      foreach $e \in \chi_o[k]$ **do**
        $W'[e] \longleftarrow W'[e] \times$ `Random`$(\gamma_o^{min}, \gamma_o^{max})$
      foreach $e \in E' - \chi_o[k] - \tau_k^c$ **do**
        $W'[e] \longleftarrow W'[e] \times$ `Random`$(1, 1.5)$
      $\chi^c[k] \longleftarrow$ `ComputeSteinerTree`(G', E', W', s_k, T_k)
  else
    $\chi^c \longleftarrow$ `RandomChoice`(χ^a, χ^b)

don't remove links as in GA for embedding, hence there is no need for "breaking down the graph into components, reconnection and removal of unused branches" as in GA for embedding. Nor is the k-shortest path algorithm needed here, because the randomized link cost multipliers substitute the necessity for a k-shortest path algorithm.

9. Performance evaluations for embedding

The objectives of this performance analysis is to show that our methods for virtual link embedding clearly succeed in capturing three essential aspects of wireless links : (1) their broadcast nature which should be exploited whenever possible when embedding point-to-multipoint virtual links; and (2) interference between neighboring links which should be avoided whenever possible; and (3) to achieve a decent load-balancing of clique and flow table utilization to improve admissibility. It also compares both proposed methods and investigates the trade-off raised by these latter: accuracy versus computation time. Below, we describe our simulation model, the main performance metrics and some of the obtained results.

9.1. Heuristic algorithms for comparison

The considered heuristic algorithm for comparison is presented in Algorithm 7. It is a simple Steiner tree construction for each VL in a request sequentially, with the two complementary specificities: (1) For each VL k in a request K, a function called updateProhibitiveLinkCost (Algorithm 7 - Line 8) is called which gives infinity link cost to links with one node that has no OpenFlow table entry left, or those links that has not enough bandwidth for the forthcoming VL k_{next}; (2) Three different link metrics are used for comparison (Algorithm 7 - Line 9), which can contribute differently to the acceptance rate:

- Metric-1: Dynamic Link Metric, as presented in Algorithm 4.
- Metric-2: Link metric (associated to the clique c) defined as $\alpha_1 + \alpha_2 + \beta_1 \left|E(c)\right|$.
- Metric-3: Link metric (associated to the clique c) defined as $\frac{\alpha_1+\alpha_2+\beta_1\left|E(c)\right|}{residual\_capacity(c)}$.

We can see that the Metric-2 and Metric-3 do not take into consideration the switching resources. It is expected that Metric-1 and Metric-3 would lead to a decent load balancing between cliques, which should not be the case for Metric-2. It is also expected that Metric-1 would help in balancing switch resource consumption.

9.2. Network Model

For space reasons, one single network instance is considered in the presented results. It is composed of 20 nodes connected via 60 links. Nodes are equipped with up to 3 radio interfaces that operate on 6 disjoint frequency bands (channels). The capacity of each channel is set to 180 units of bandwidth (UB). The left side of Figure 10 depicts the network topology, each link color reflects a frequency band. It leads to 9 cliques (as depicted in right side of Figure 10) with a number of members ranging from 3 to 11 links. Unless specified, the flow table and group table maximum size are set to 1000 and 100, respectively. In fact, this network instance can been seen as derived

Algorithm 7: Shortest Path Heuristic for Comparison

Input : $G(V, E); \vec{\mathbf{K}}$;
Output: $\chi = [\tau_1, \tau_2, ..., \tau_{|K|}]\}$

```
begin
    G'(V', E') ⟵ Clone(G(V, E))
    W = InitiateLinkCosts()
    foreach K ∈ K⃗ do
        foreach k ∈ K do
            τ_k ⟵ ComputeSteinerTree(G', s_k, T_k, W)
            χ[k] ⟵ τ_k
            updateProhibitiveLinkCost(G', W)
        updateLinkCost(G', W, χ)
```

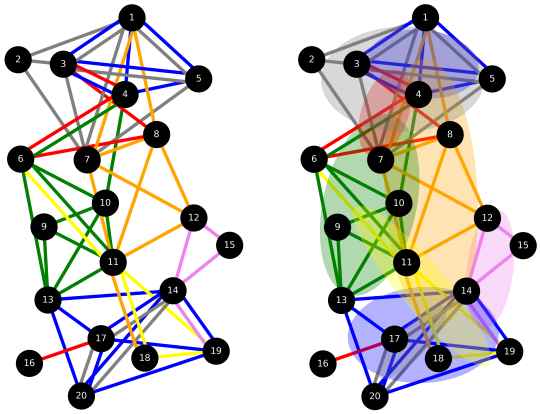

Figure 10: Network model used in our performance evaluation. Instead of using the usual way of presenting cliques with conflicting links as "node", cliques are directly drawn on the illustrating graph, with ellipsoids covering the midpoint of each link in the clique.

from our test-bed of software-defined wireless multi-hop network, with most nodes equipped with three 802.11 WiFi radio interfaces (multi-radio), each able to operate on either one of the three separated channels of the ISM 2.4GHz band or one of the UNII 5GHz channels, shown in [2].

Hereafter we present how we can calculate the conflict graph, which serves then as an input for our embedding algorithms, given the positions of the 20 nodes in a 2-D xy-axis space. First, we implicitly assume a free space propagation model for all transmissions, two links operating on the same channel interfere if (1) they have one node in common or (2) one transmission on a link has an RSSI (Received Signal Strength Indicator) at one of the nodes of the other link, which is greater than a predefined threshold. The relationship between the distance and RSSI is determined by the logarithmic wireless path model:

$$\mathrm{RSSI}(d) = \mathrm{RSSI}(d_0) + 10\eta \log(d/d_0) + \zeta$$

where d is the distance, d_0 is the distance of reference, η is the path attenuation factor and ζ is the shadowing factor. Therefore, towards the end, in the interference model used for this performance analysis, it is the position of nodes that determines the conflict graph and hence cliques.

9.3. Load Model

Virtual links requests are composed of a number of point-to-point and point-to-multipoint virtual links randomly chosen between 4 and 6. Each point-to-multipoint virtual link has a number of destinations randomly chosen between 2 and 6. The bandwidth requirement of each virtual link is also chosen randomly from 1 to 3 UB. Source and destination selection is performed on a random basis. The request arrivals follow a Poisson process with an arrival rate r of 0.01, 0.02, 0.03, 0.04, i.e. in average 1, 2, 3 or 4 requests each 100 units of time (UT). The request life-time conforms to an exponential distribution with an average of 1000 UT.

9.4. Simulation Settings and Implementation

The Integer Linear model was implemented in Python with CPLEX 12.63 solver. The experiments were carried out on a virtual machine with 25 vCPU and 16GB of RAM and running Ubuntu 14.04. A gap of less than 1% to the optimal solution is considered satisfactory. Unless specified, path splitting is disabled for ILP. For GA, the implementation is in Python (running on pypy [1]) using deap [31]. Unless specified, population size is set to 18 and number of generations at 18. Cross-over probability is 0.9 and mutation probability is 0.05. N_{top} is set to 2. The simulation horizon is fixed to 10000 UT (this time period is sufficient to have our methods in the stationary regime). α_1, α_2 and α_3 are set to 1, 1 and 5 respectively throughout all evaluation experiments.

NetworkX [32] is used to represent graphs. It is to note that the discovery of cliques within a conflict graph (which is NP-hard) can be easily conducted via its find_cliques API [2], which is based on on the algorithms published in [33] [34] and discussed in [35]. For persistence and serialization of graph models , network models and request models, dill [36] is used.

9.5. Performance metrics

The following performance metrics are computed during simulation for performance analysis purposes:

- Acceptance rate (ac, in %): the percentage of successful virtual links requests out of all the requests that arrived during the simulation time or accumulatively with the time.
- Clique utilization (cu, in %): bandwidth allocated at the links composing a clique divided by channel capacity, computed as $100 * \frac{\xi'(c)}{B_\lambda}$.
- Switch resource utilization regarding flow table utilization (su, in %): flow table utilization at the nodes divided by the initial flow table size, computed as $100 * \frac{L'_v}{L_v}$.

[1] http://pypy.org/

[2] https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.clique.find_cliques.html

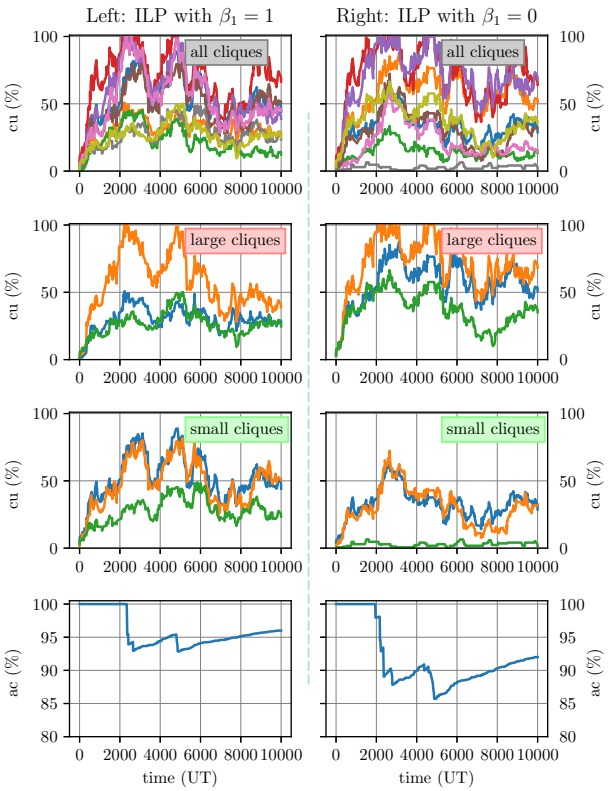

Figure 11: Clique utilization (all cliques, large cliques and small cliques) and acceptance rate with $\beta_1 = 1$ v.s $\beta_1 = 0$. Computed with ILP. Arrival rate = 0.02. $\beta_2 = 0, \beta_3 = 15$. Similar results are obtained with GA.

- Switch resource utilization regarding group table utilization (gu, in %): group table utilization at the nodes divided by the initial group table size, computed as $100 * \frac{M'_v}{M_v}$.
- Computation time (in second): the average computation time for one request.

9.6. Performance Results

9.6.1. Coping with wireless links interference

The objective is to assess how efficient are our methods in reducing and avoiding wireless links interference. To this end, β_2 and β_3 are set to 0 in a first place. We compare the effect of setting β_1 to 1 versus to 0.

In fact, when embedding virtual links requests, our methods favor links belonging to cliques with limited number of members, introducing, by the way, in their surroundings less interference and, hence, preserving the overall available bandwidth. This is clearly shown in Figure 11 which focuses on the clique utilization of two groups of cliques: small cliques with a small number ($3 \sim 6$) of interfering links and large cliques with a high number ($8 \sim 11$) of links. When activating interference reduction, the bandwidth consumed by large cliques is decreased contrary to small cliques. As expected a portion of the bandwidth consumed by a large-size clique is transferred to smaller-size cliques: small cliques experience an increase in clique utilization while large cliques get less loaded.

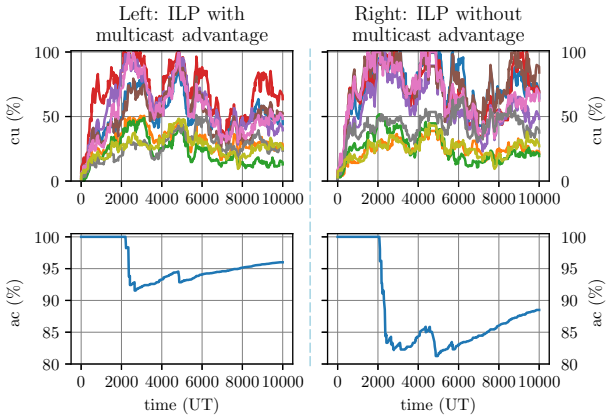

Figure 12: Clique utilization and acceptance rate with and without multicast advantage. Computed with ILP. Arrival rate = 0.02. $\beta_2 = 0, \beta_3 = 15$. Similar results are obtained with GA.

Favoring small-size cliques may lead to longer data paths and hence more resources are needed to support the virtual links being embedded. Since the acceptance rate is improved with such a strategy, this means that this latter increase is compensated by the resource that are preserved thanks to interference reduction. With the considered network and load models, our experiments show a slight increase around 1% on the average length of selected data paths.

9.6.2. Assessing the gain brought by the Multicast advantage

The objective is to quantify the gain in resource usage that our methods achieve by exploiting the multicast advantage when embedding point-to-multipoint virtual links. Again, β_2 and β_3 are set to 0 in a first place. β_1 is set to 1 as we have shown that interference should be taken into consideration for the modeling.

The clique utilization of the 9 cliques is presented in Figure 12. We see that disabling the multicast advantage induces extra bandwidth consumption that overloads all cliques and causes significantly more embedding failures.

9.6.3. Clique utilization balancing

By setting β_2 to 5, we activate clique load balancing. To show the effect of clique load balancing, Figure 13 shows that the clique utilizations have now much less disparity, with ILP as well as GA, compared to Figure 11 and 12 where $\beta_2 = 0$, leading to an improved acceptance rate (99.5% for ILP and 98.5% for GA) .

9.6.4. Switch resource consumption and balancing

To show how flow table resource is consumed and balanced and in which manner this might impact , we set the initial flow table size to 90, and compared the results of $\beta_3 = 0$ versus $\beta_3 = 15$ using ILP, as is shown in Figure 14. Apart from a much better balancing of flow table resource utilization, we also see an improved acceptance rate (99.5% v.s. 94.5%). Hence, flow table resource balancing should be activated. The effect of switch resource balancing of GA is shown in Figure 15. We

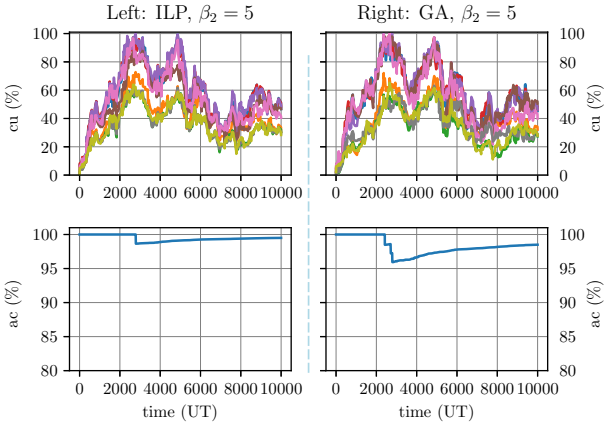

Figure 13: Clique utilization balancing with ILP and GA. Arrival rate = 0.02. β_3 = 15.

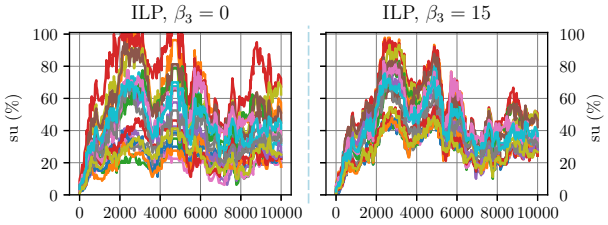

Figure 14: Switch resource consumption of all nodes, computed with ILP, with initial flow table size set at 90. Arrival rate = 0.02. β_2 = 5

can see that GA is less effective than ILP in switch resource balancing.

Figure 16 presents the group table utilization, computed with ILP and GA. We observe that with ILP and GA, thanks to the multicast advantage, the group table consumption remains very limited despite the successful mapping of point-to-multipoint virtual links. As expected, for the considered simulation model, group table entries are abundant in comparison to embedding needs. As a consequence, it does not play a decent role in the selection of the data paths. Hence, there is no need to balance its utilization in the formulation.

9.6.5. *Path splitting*

Figure 17 shows that with path splitting, the acceptance rate improves slightly. However, if we limit the initial switch resources, then we observe that ILP-PS delivers much worse results than ILP and GA. Figure 20 depicts this with initial flow table size set to 90. We see that ILP-PS consumes more group table entries and flow table entries, and the latter will lead to embedding failures of ILP-PS compared to ILP.

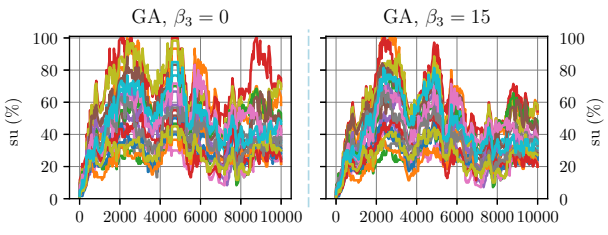

Figure 15: Switch resource consumption of all nodes, computed with GA, with initial flow table size set at 90. Arrival rate = 0.02. β_2 = 5.

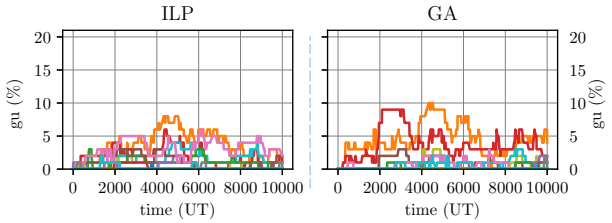

Figure 16: Group table consumption of ILP and GA. Arrival rate = 0.02. β_2 = 5, β_3 = 15.

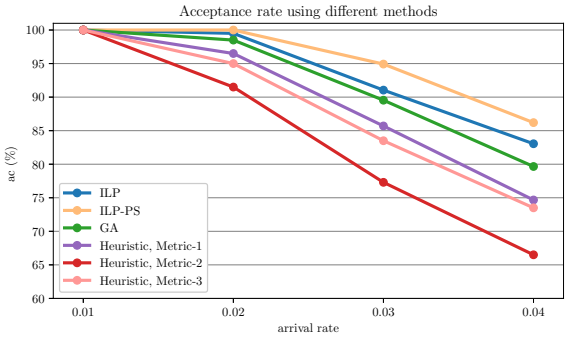

Figure 17: Acceptance rate of the heuristic using 3 different link metrics as well as ILP, GA and ILP-PS, for different arrival rates. β_2 = 5, β_3 = 15.

9.6.6. *Comparison with heuristic algorithms*

Here we present the results of SPH using different link metrics, as is shown in Figure 17. We see that the heuristic with all 3 link metrics has a lower acceptance rate than ILP, ILP-PS and GA. However, Metric-1 has the best performance, while Metric-2 has the worst. Also, Figure 18 and Figure 19 show that Metric-1 leads to a relatively good balancing both in clique load and switch resource utilization, while Metric-3 leads only to a decent balancing in clique load. Metric-2, as expected, leads to an unbalanced situation, and hence gives the worst acceptance rate.

9.6.7. *Embedding method selection : ILP v.s. GA*

There are two important criteria to consider when choosing the method to be applied: (1) accuracy (leading to optimal

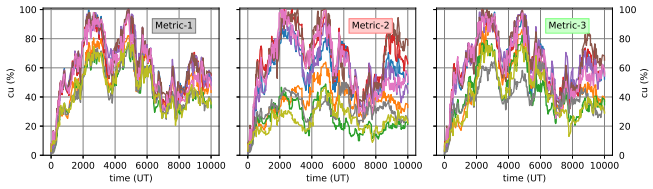

Figure 18: Clique load balancing of heuristic using the 3 different link metrics

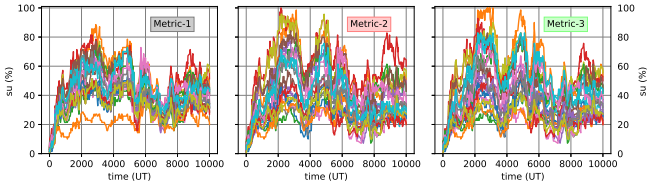

Figure 19: Switch resource balancing of heuristic using the 3 different link metrics

acceptance rate) and (2) computation time. Figure 17 shows the acceptance rate using different methods, and we can see that ILP-PS (ILP with path splitting enabled) gives slightly better results than ILP and GA. Figure 21 and Figure 22 present the acceptance rate and the computation time obtained with GA when considering different population and generation sizes. For the considered network model, GA with a population of 18 individuals and 18 generations lasts 80% of the computation time of ILP. With smaller population and generation size (e.g. 12 and 12), the computation time can be significantly reduced (less than 1/10 of ILP), bringing only minor degradation of the acceptance rate (~1.5%). Our experiments show that for larger network models, GA shows a significant advantage in computation time compared to ILP. For example, if we take a network substrate composed of of 60 nodes and 166 cliques using 7 channels (shown in Figure 23), our experiments show that with the graph as input, it takes ILP more than 4 minutes to compute the results for each virtual link request. As a contrast, it takes GA (with a population size of 18 and a generation number of 18) less than 1 minute. Our experiments for networks composed of up to 100 nodes confirm this trend. On the contrary, with ILP-PS, as the search space explodes, the computation time can be several folds of that of ILP and hence much more than GA. This is another major shortcoming of ILP-PS, besides more induced consumption of switch resources.

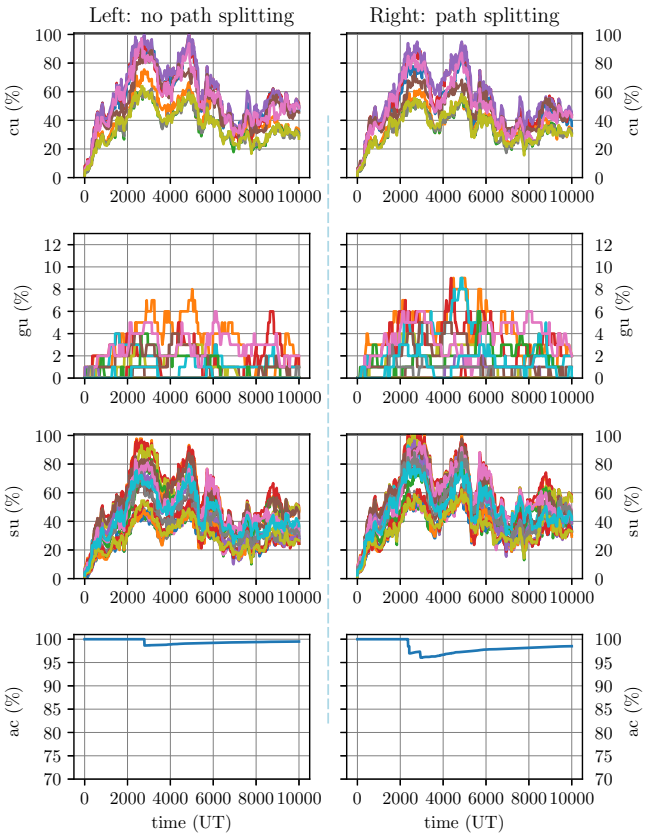

Figure 20: Switch resource consumption with ILP and ILP-PS. The initial flow table size is set to 90. Arrival rate = 0.02. $\beta_2 = 5, \beta_3 = 15$.

10. Re-embedding vs embedding for changing VLs

When a VL evolves, the data paths on which it is established need to be recomputed in order to keep providing the requested QoS. These data paths can be recomputed from scratch by using the embedding algorithm. This leads to minimal network resource consumption but typically implies VL service disruption, the time the newly computed data paths get installed in place of the initial ones.

Another alternative followed by our re-embedding algorithm is to rely on the already provisioned data paths and proceed with some additions and adjustments. The main expected benefit is a limited service disruption time with a price to pay in terms of a less efficient (near optimal) network resource consumption.

Therefore, the objective is to compare the re-embedding to the embedding algorithm in terms of service disruption and consumed resources, when used for changing VLs.

To that end, we adopt the network model, load model and simulation settings of the previous section, and we consider a scenario where the network is not highly loaded. Indeed, it is under

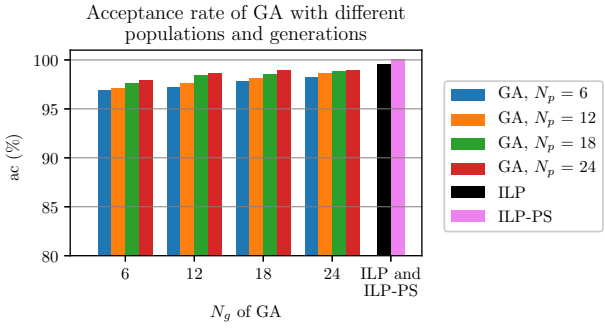

Figure 21: Acceptance rate of GA by varying number of generation and size of population, compared with ILP and ILP-PS. Arrival rate = 0.02

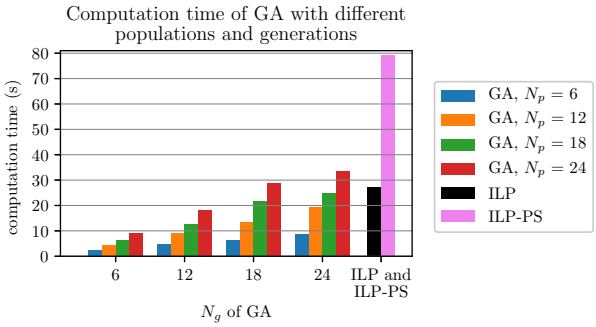

Figure 22: Average Computation Time of each request of GA by varying number of generation and size of population, compared with ILP and ILP-PS.

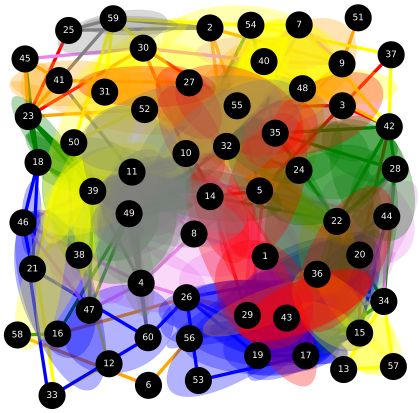

Figure 23: A large network substrate composed of 60 nodes and 166 cliques, using 7 channels

this condition that it should be used to limit service disruption at the cost of a slight increase in network resources consumption (in comparison to the optimal solution). More precisely, the initial capacity of each wireless channel at 500 UB, and the initial flow table and group table sizes are respectively set to 5000 and 500. The arrival rate is set at 0.02. Each time a new request K_o arrives, the embedding algorithm is used (ILP as defined in Section 5, with $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 5, \beta_1 = 1, \beta_2 = 1, \beta_2 = 5, \beta_3 = 15$, path splitting disabled). For every VL $k_o \in K_o$, at its mid-life, a random number between 1 and 4 of new destinations are added, forming an evolving sub-request $k_e \in K_e$. To treat request K_e, we use the embedding algorithm of Section 5, and the re-embedding algorithm of Section 6 with $\rho = 100000$ (similar results are observed with the genetic algorithms of Sections 7 and 8).

Our main findings are the following. With the considered assumptions no service disruption is observed with our re-embedding algorithm. When the embedding algorithm is used, in more than 42% of the cases, the VLs requests are embedded on different data paths leading to service disruptions. Regarding the consumed network resources, in comparison to the embedding algorithm, the near optimal data paths computed by the re-embedding algorithm (that avoid service disruption) induce an average increase in bandwidth resource consumption of 5.4% and switching resource consumption of 4.7%. A detailed analysis of the data paths computed by the re-embedding algorithm reveals that the "multicast advantage" is often adopted. This is the main reason why the increase in network resource consumption remains limited. This is fostered by the highly meshed topology of the considered wireless network.

11. Conclusion

In this paper, we developed an Integer-Linear programming method and a genetic algorithm method for the resource allocation of multiple virtual links in wireless software defined multi-radio multi-channel multi-hop networks. In comparison to existing works, the main contribution of our proposals lies in the conjunction of the following features: (1) the support of point-to-multipoint virtual links in addition to point-to-point virtual links, and, in a wireless context, how to benefit from the multicast advantage to gain in bandwidth consumption, (2) the consideration of switching resources in the allocation of resources in addition to the bandwidth of channels. Through our evaluations, we show that both of our proposed methods work well. More interestingly, we investigated how the consideration of interference, multicast advantage as well as resource balancing could impact the embedding results, and, how the two proposed methods differ in performance and computation time. We have also proposed two re-embedding algorithms which can be incorporated into our scheme to support dynamic VLs requests with the objective of limiting, if not avoiding whenever possible, service disruption.

One important perspective to this work is to extend our algorithms by considering more general interference models, more particularly, those which capture the cumulative interference caused by transmissions form multiple nodes, e.g. the SINR interference model. Another future direction is to consider two other QoS requirements, namely end to end transfer delay and reliability.

References

[1] I. T. Haque and N. Abu-Ghazaleh,"Wireless Software Defined Networking: A Survey and Taxonomy," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2713-2737, Fourthquarter 2016.

[2] Lunde Chen, Slim Abdellatif, Pascal Berthou, Kokouvi Benoit Nougnanke, and Thierry Gayraud. "A Generic and Configurable Topology Discovery Service for Software Defined Wireless Multi-Hop Network". In *Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access (MobiWac '17)*, 2017

[3] H. C. Yu, G. Quer and R. R. Rao, "Wireless SDN mobile ad hoc network: From theory to practice," *IEEE International Conference on Communications (ICC)*, Paris, 2017, pp. 1-7.

[4] A. De Gante, M. Aslan and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," *27th Biennial Symposium on Communications (QBSC)*, Kingston, ON, 2014, pp. 71-75.

[5] Won Jin Lee, Jung Wan Shin, Hwi Young Lee and Min Young Chung, "Testbed implementation for routing WLAN traffic in software defined wireless mesh network," *International Conference on Ubiquitous and Future Networks (ICUFN)*, Vienna, 2016, pp. 1052-1055.

[6] D. Gomez-Barquero, D. Navratil, S. Appleby and M. Stagg, "Point-to-Multipoint Communication Enablers for the Fifth Generation of Wireless

Systems," in IEEE Communications Standards Magazine, vol. 2, no. 1, pp. 53-59, MARCH 2018. doi: 10.1109/MCOMSTD.2018.1700069 keywords

[7] M. Li, C. Hua, C. Chen and X. Guan, "Application-driven virtual network embedding for industrial wireless sensor networks," *IEEE International Conference on Communications (ICC)*, Paris, 2017, pp. 1-6.

[8] S. Abdelwahab, B. Hamdaoui, M. Guizani and T. Znati, "Efficient Virtual Network Embedding With Backtrack Avoidance for Dynamic Wireless Networks," in *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2669-2683, April 2016.

[9] Donggyu Yun, et al., "Embedding of virtual network requests over static wireless multihop networks", *Computer Networks*, Volume 57, Issue 5, 2013, pp 1139-1152

[10] Pin Lv, Xudong Wang, and Ming Xu. "Virtual access network embedding in wireless mesh networks". *Ad Hoc Netw.* 10, 7 (September 2012), 1362-1378.

[11] G. Di Stasi, S. Avallone and R. Canonico, "Virtual network embedding in wireless mesh networks through reconfiguration of channels," *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Lyon, 2013, pp. 537-544.

[12] P. Lv, Z. Cai, J. Xu and M. Xu, "Multicast Service-Oriented Virtual Network Embedding in Wireless Mesh Networks," in IEEE Communications Letters, vol. 16, no. 3, pp. 375-377, March 2012.

[13] M. Li, C. Chen, C. Hua and X. Guan, "Intelligent Latency-Aware Virtual Network Embedding for Industrial Wireless Networks," in *IEEE Internet of Things Journal*, 2019

[14] H. Afifi, S. Auroux and H. Karl, "MARVELO: Wireless virtual network embedding for overlay graphs with loops," 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, 2018, pp. 1-6.

[15] Afifi, Haitham, and Holger Karl. "Power Allocation with a Wireless Multi-cast Aware Routing for Virtual Network Embedding." in 16th IEEE Annual Consumer Communications & Networking Conference (CCNC). IEEE, 2019.

[16] H. Jmila and D. Zeghlache, "An adaptive load balancing scheme for evolving virtual networks," 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, 2015, pp. 492-498. doi: 10.1109/CCNC.2015.7158024

[17] P. N. Tran and A. Timm-Giel, "Reconfiguration of virtual network mapping considering service disruption," 2013 IEEE International Conference on Communications (ICC), Budapest, 2013, pp. 3487-3492. doi: 10.1109/ICC.2013.6655090

[18] Y. Zhu and M. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, Barcelona, 2006, pp. 1-12. doi: 10.1109/INFOCOM.2006.322

[19] Z. Cai, F. Liu, N. Xiao, Q. Liu and Z. Wang, "Virtual Network Embedding for Evolving Networks," 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, Miami, FL, 2010, pp. 1-5. doi: 10.1109/GLOCOM.2010.5683160

[20] L. Chen, S. Abdellatif, T. Gayraud and P. Berthou, "A Steiner tree based approach for the efficient support of multipoint communications in a multi-domain context," *2017 IEEE Symposium on Computers and Communications (ISCC)*, Heraklion, 2017, pp. 316-321.

[21] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of Interference on Multi-hop Wireless Network Performance," *ACM Mobicom*, San Diego, CA, USA, September 2003

[22] X. Cheng, P. Mohapatra, S.-J. Lee, S. Banerjee, "MARIA: Interference-Aware Admission Control and QoS Routing in Wireless Mesh Networks", IEEE International Conference on Communications, 2008

[23] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in Proc.ACM MobiCom, 2006, pp. 227–238.

[24] R. Gupta, J. Walrand, "Approximating maximal cliques in ad-hoc networks", Proceedings of IEEE 15th International Symposium on Personal, Indoor and Mobile Radio Communications,2004

[25] J. Musacchio, R. Gupta, J. Walrand. "Sufficient Rate Constraints for QoS Flows in Ad-Hoc Networks". *INFOCOM*, 2005

[26] F. Cazals, C. Karande, "A note on the problem of reporting maximal cliques", *Theoretical Computer Science*, Volume 407, Issues 1-3, 6 November 2008, Pages 564-568

[27] P. Gupta and P. R. Kumar, "The capacity of wireless networks," IEEE Transactions on Information Theory, vol.46, no. 2, pp. 388–404, 2000.

[28] M. Capelle, S. Abdellatif, M. J. Huguet and P. Berthou, "Online virtual links resource allocation in Software-Defined Networks," *IFIP Networking Conference (IFIP Networking)*, Toulouse, 2015, pp. 1-9.

[29] T. Lu and J. Zhu, "Genetic Algorithm for Energy-Efficient QoS Multicast Routing," in *IEEE Communications Letters*, vol. 17, no. 1, pp. 31-34, January 2013.

[30] M. Parsa, Qing Zhu and J. J. Garcia-Luna-Aceves, "An iterative algorithm for delay-constrained minimum-cost multicasting," in *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 461-474, Aug 1998.

[31] François-Michel De Rainville, Félix-Antoine Fortin et al., "DEAP: a python framework for evolutionary algorithms". In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation (GECCO '12)*, 2012

[32] Hagberg, Aric, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using NetworkX. No. LA-UR-08-05495; LA-UR-08-5495. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[33] Bron, C. and Kerbosch, J. "Algorithm 457: finding all cliques of an undirected graph". Communications of the ACM 16, 9 (Sep. 1973), 575–577

[34] Etsuji Tomita, Akira Tanaka, Haruhisa Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments", Theoretical Computer Science, Volume 363, Issue 1, Computing and Combinatorics, 10th Annual International Conference on Computing and Combinatorics (COCOON 2004), 25 October 2006, Pages 28–42

[35] F. Cazals, C. Karande, "A note on the problem of reporting maximal cliques", Theoretical Computer Science, Volume 407, Issues 1–3, 6 November 2008, Pages 564–568

[36] M.M. McKerns, L. Strand, T. Sullivan, A. Fang, M.A.G. Aivazis, "Building a framework for predictive science", Proceedings of the 10th Python in Science Conference, 2011