Computing end-to-end QoS Paths in the Internet Considering Multiple Alliances

Romain Jacquet, Géraldine Texier, Alberto Blanc

▶ **To cite this version:**

Romain Jacquet, Géraldine Texier, Alberto Blanc. Computing end-to-end QoS Paths in the Internet Considering Multiple Alliances. Networks 2014 : 16th International Telecommunications Network Strategy and Planning Symposium, Sep 2014, Funchal, Portugal. pp.1 - 6, 10.1109/NETWKS.2014.6959248 . hal-01167357

HAL Id: hal-01167357
https://hal.science/hal-01167357v1

Submitted on 24 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Computing end-to-end QoS Paths in the Internet Considering Multiple Alliances

Romain Jacquet*, Géraldine Texier*, Alberto Blanc*
*Telecom Bretagne/Irisa, 2 rue de la châtaigneraie, Cesson Sévigné, 35576, France
Email: {firstname.lastname}@telecom-bretagne.eu

***Abstract*—Value added services like VoIP, videoconferencing and IPTV need end-to-end Quality of Service (QoS) guarantees in order to work correctly. As the Internet is a collection of Autonomous Systems (AS), most of the time the communication endpoints belong to different ASes, so that all the ASes traversed by the communication must cooperate in order to offer end-to-end guarantees. Yet each AS is usually unwilling to disclose any detail about its internal network. To address this confidentiality issue we propose a system where each AS publishes a list of offers, specifying the QoS guarantees between its entry and exit points, without specifying anything else about its internal network. As proposed in several works, it is also possible for ASes to form alliances, which can be seen as "macro ASes" that publish the available offers between the entry and exit points of the alliance. In this paper we present ACQA, an algorithm that can find end-to-end paths satisfying given QoS constraints by combining the offers of several alliances and/or ASes.**

I. Introduction

In the Internet, value-added services (HD video, online games, cloud computing) represent an ever increasing share of the total traffic. In order to achieve their full potential, these services need end-to-end Quality of Service (QoS) guarantees on several parameters like bandwidth, delay, and packet loss. Several studies have addressed the problem of finding paths satisfying a set of given requirements. This problem is also known as multi-constrained routing, which is known to be NP complete [1].

Practical solutions to this problem must be scalable, given the size of the Internet, and must also allow each Autonomous System (AS) to keep confidential its internal structure. The latter problem can be addressed by combining several Service Level Agreements (SLA), one for each AS. An SLA is a contract, where each AS specifies what guarantees it can offer between an entry and an exit point. As these guarantees apply only within a single domain, one must combine several SLAs in order to offer end-to-end guarantees. For such a solution to work, each AS needs only to publish a list of SLAs that it can offer between its entry and exit points. It does not need to disclose how these guarantees are implemented, meeting the confidentiality requirement. Moreover, each AS is free to choose different technical solutions, respecting its autonomy with respect to other ASes.

As the number of ASes in the Internet is large (about 50000 at the time of this writing [2]), and as some of them have non-overlapping geographical coverage, it is conceivable that a certain number of ASes could form alliances (or federations). Several works have already proposed different variants of such a scheme [3]–[7]. We define an alliance as a group of ASes that trust each other and that agree to share business and/or technical policies, similarly to what has been proposed by the ETICS project [4]. Given the number of ASes in the Internet we conjecture that they will create more than one alliance, with each AS belonging to only one alliance and with some ASes remaining independent (i.e., they do not belong to any alliance).

Unlike previous works that deal with finding QoS paths within a single alliance, we introduce ACQA an algorithm capable of finding end-to-end QoS paths involving several ASes and/or alliances. This is an extension of the SANP algorithm [8], which can find feasible non-dominated paths in a graph comprised only of ASes (i.e., no alliances). ACQA outperforms SANP as the paths it finds tend to dominate the paths found by SANP (based on the generational distance [9] and zitzler [10] metrics).

The remainder of the paper is organized as follows: we present our model in Section II and related works in Section III. In Section IV, we introduce ACQA and in Section V we present and discuss the simulation results.

II. Model

A. The Internet AS Graph

The Internet topology is often modeled as a graph $G(V, E)$ where V is a set of vertices representing the ASes and E is a set of edges representing the inter-domain links. We assume that each AS is willing to publish a list of available SLAs, describing the QoS guarantees it can offer between two of its Autonomous System Border Routers (ASBRs). Each SLA lists the

guarantees offered on a certain number of metrics (e.g., bandwidth, delay, packet loss) for each pair of entry-exit point, and the corresponding price. Note that an AS can decide not to offer any guarantees between a certain number of ASBRs by publishing an SLA advertising zero capacity as well as infinite delay and packet loss between these ASBRs, in other words the AS can offer only a best effort service between some of its ASBRs. As long as these best-effort-only SLAs are the exception and not the norm, it will be possible to offer end-to-end QoS guarantees, in most cases, thanks to the great variety of paths in the Internet.

We represent an AS by a complete graph in which all the interfaces of its ASBRs are connected. These intra-domain links represent the SLAs offered by the ASes. It is important to note that they do not represent at all the internal topology of the ASes. Hence, the confidentiality property of the ASes is respected. We connect each interface of the ASBR to exactly one interface of an ASBR belonging to a different AS. These edges represent the Inter-domain links. We characterized each intra-domain link by a vector w with K weights representing the K additives QoS metrics. For simplicity reasons, we assume the QoS metrics to be the same in both directions so that we use an undirected graph. Figure 1 shows a sample topology of six ASes with the corresponding ASBRs. While the constraint of having each ASBR connected to only one ASBR in a different AS might seem a limitation of the model, it can be easily addressed by using multiple ASBRs to represent any ASBR that is connected to more than one ASBR belonging to another domain. In other words, one can think of the ASBR as actually representing the interfaces of the router.

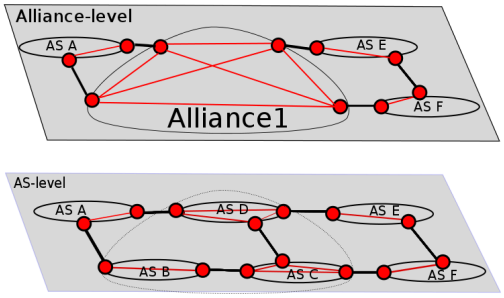

Fig. 1: Representation of the AS-offers and alliance-offers within the Internet

B. Alliances

Starting from a graph $G(V, E)$ as defined above (Section II-A), it is possible to construct an Alliance-graph $G'(V', E')$ in which the nodes represent either ASes or alliances. In the same way the ASes publish offers (AS-offer) between the interfaces of its ASBRs, we assume that alliances publish offers (alliance-offers) between the edge-interfaces of its edge-ASBRs. An edge-interface is an interface connected with another interface that belongs to an AS outside the alliance. An edge-ASBR is an ASBR that has at least one edge-interface. For rest of the paper we denote an edge-interface of an edge-ASBR as an edge-ASBRI. We connect by a complete graph the edge-ASBRIs of an alliance.

These intra-alliance links represent the offers (SLAs) of the alliances, hiding the internal topology for confidentiality purposes. We characterize each intra-alliance link by a vector w with K weights representing the K additives QoS metrics associated to the SLA offered between the linked nodes. Just like each AS is free to decide the SLAs it offers, it is up to each alliance to define a list of available SLAs. The ASes belonging to the alliance do not need to specify how these alliance-SLAs are implemented inside the alliance. As an example, let's assume that the ASes B, C and D in the AS-level in Figure 1 decide to form an alliance named Alliance1. The corresponding Alliance-level topology is shown in the top layer in Figure 1.

III. Related Works

Several works have already considered the notion of alliance between different ASes. Most of them focuses on finding QoS paths within a single alliance. The FP7 project ETICS [4] introduced different notions of alliance, depending on the trust between the members, the business and/or technical policies that are shared to a lesser degree. For example, in an *open association* (lower level of trust), only bilateral agreements between neighbors (SLAs exchanges) are allowed. It is possible to build end-to-end paths by combining the SLAs of the ASes along a path.

In [5] the authors propose to compute QoS paths within an alliance via the Routing Control Platform (RCP) [11]. RCP is an architecture that aims at solving the scalability issue of internal Border Gateway Protocol in a large full mesh network. RCP has been designed to correctly distribute the routes in fast and reliable way among the routers. In the context of computing QoS paths, the authors utilize the RCP architecture to install a RCP entity within each domain of an alliance. The RCP entities are responsible for establishing and maintaining connections between each other and to compute and to handle connection requests with QoS requirements. To do so, they exchange reachability information through TCP.

In [6], *Service Level Specifications* (SLS) must be published to a neutral centralized third party, which is responsible for negotiating, on behalf of the customer, a chain of SLSs with the rest of the members such that the

customer request is satisfied. In [7], the authors use a Q-learning algorithm to negotiate SLAs between NSPs that belong to the same federation. In [12], the authors present the notion of *federation* as a short association between different stakeholders within the Internet in order to achieve end-to-end services.

To the best of our knowledge, [3] is the only work that considers several alliances when searching for end-to-end QoS paths. They do not consider the notion of "alliance-offer" but rather, an alliance is defined as a group of ASes that share some information about network services availability and reachability. User requests specify only the source node, the requested service and QoS constraints but not the destination node. In our work we consider communications between *two* given nodes and not between a node and any other node capable of offering a certain service.

IV. Algorithm

We present ACQA an algorithm capable of finding end-to-end QoS paths within a graph composed of ASes and alliances. The basic idea of ACQA is to construct a sub-graph covering a limited region around the shortest path from the destination to the source in order to identify feasible non-dominated paths that can satisfy a connection request. The algorithm relies on the notion of neighborhood, i.e., the set of nodes whose distance from a reference node is at most r_0. More precisely, as explained in section II-B, let $G' = \{V', E'\}$ be the AS and alliance-level graph. The neighborhood of node i is the set $N_i = \{V'_i, E'_i\}$ where $V'_i = \{x | d(i,x) \leq r_0, x \in V\}$ and $E'_i = \{e | e(j,k) \in E, j \in V'_i, k \in V'_i\}$ where $d(i,j)$ is the distance (shortest path) between node i and node j and $e(j,k)$ is the edge between nodes j and k. We suppose that each node i in the graph knows its neighborhood N_i.

A source node n_s wishing to find an end-to-end path with destination node n_d and with given QoS guarantees will send a request q specifying the QoS constraints to n_d. The request is forwarded to n_d through the existing routing mechanism. Upon the reception of this request, n_d will execute the algorithm in Figure 2. For the sake of simplicity, throughout the paper, we are going to assume that this existing routing mechanism uses the shortest path even though ACQA works with any other routing criteria.

The execution of the algorithm starts at the destination (line 1), by initializing the variables T and G'' (lines 2, 3) that will contain, respectively, the nodes in the shortest path *from* the destination *to* the source and the sub-graph. The loop starting at line 4 is repeated at each node along the shortest path: the current node (n) adds itself to the set T and then merges its neighborhood with the sub-graph (line 6). Recall that (V'_n, E'_n) is the

ACQA(q, n_s):

1: $n \leftarrow n_d$ {phase 1}
2: $T \leftarrow \emptyset$ {set of the nodes already visited, it will contain the shortest path between n_s and n_d}
3: $G'' \leftarrow (V'' \leftarrow \emptyset, E'' \leftarrow \emptyset)$ {G'' is the (initially empty) sub-graph}
4: **repeat**
5: $T \leftarrow T \cup \{n\}$
6: $G''(V'', E'') \leftarrow G''(V'' \cup V'_n, E'' \cup E'_n)$ {merge the neighborhood of node n to the sub-graph}
7: **if** $|V''| > M$ **then**
8: $G'' \leftarrow$ limitGraphSize(G'',M,T) {heuristic to reduce the size of G''}
9: **end if**
10: $n \leftarrow n$.nextAS(n_s)
11: n.send(G'', q) {the request and the sub-graph are forwarded to the next node (AS or alliance) on the path toward n_s}
12: **until** $n = n_s$
13: **return** n_s.selectPath(G'') {phase 2}

Fig. 2: Computation of an end-to-end path between n_s and n_d for request q.

neighborhood of node n and that each node knows its own neighborhood. Note that all these sets are defined whether n is a single AS or an alliance. The offers published by an alliance are an example of an "open association" as defined by the ETICS project: an alliance-offer is the composition (juxtaposition) of several AS-offers within the alliance and can be constructed by any of the solutions proposed in Section III. It is reasonable to consider that the alliance has several possibilities to instantiate its offers, therefore each offer is valid for a fairly long time (at least of the order of a few days, if not weeks or months) and its cost can be computed in advance. For each alliance, ACQA considers the offers between the entry ASBR and each exit ASBR connected to the following alliance (or AS) in the path. This is possible because, when ACQA is searching for paths in G'' it first finds a path in G'' and it considers the cost of traversing an alliance (or an AS) only when it already knows the next hop.

If the size of the sub-graph is above a given threshold (line 7), we use the algorithm presented in Figure 3 to reduce its size, as explained below. Once the request reaches the source, G'' is a connected sub-graph containing both the source and the destination. The source uses this sub-graph in phase 2 (line 13) to compute the feasible and non-dominated paths between itself and the destination. Figure 4 shows the resulting sub-graph G'' at the end of the first phase. In our implementation (discussed in section V), the source uses a modified depth first search to explore all the simple

limitGraphSize(G'', M, T)

1: $H \leftarrow (V_H \leftarrow T, E_H \leftarrow \{e(j,k) | j \in T, k \in T\})$
2: **while** $|V_H| < M$ **do**
3: $\quad C_j \leftarrow \{e(j,k) | k \in H\} \forall j \in \{G'' \setminus H\}$
4: $\quad l \leftarrow \arg\max_j \{|C_j|\}$
5: $\quad H \leftarrow (V_H \cup \{l\}, E_H \cup C_l)$
6: **end while**
7: **return** H

Fig. 3: Heuristic to limit the size of G'' to at most M nodes

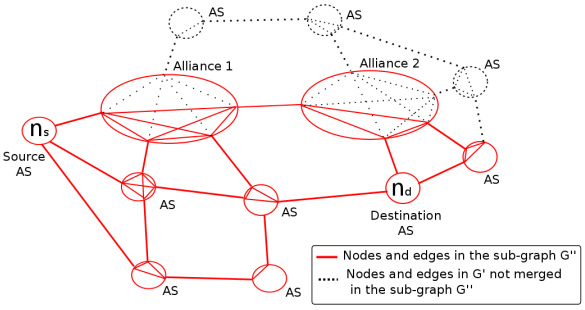

Fig. 4: AS and alliances in the computed sub-graph

paths of length at most 8 within G''. Note that the source can use other algorithms (heuristics) to compute these paths.

In order to improve the scalability of the algorithm we use a heuristic to reduce the number of nodes in the sub-graph. For each request we fix the maximum number of nodes in the sub-graph as a multiple (α) of the length of the shortest path between the source and the destination. In the implementation presented in section V, we set $\alpha = 50$. Figure 3 presents the algorithm used to ensure that the size of G'' is at most M. It starts by setting H to the portion of the shortest path explored so far (line 1). Note that as each node in T has either one or two of its neighbors in T, as it is a node on the shortest path between the source and the destination. Then (line 3) for each node j in G'' but not already in H, it builds the set C_j containing all the *edges* between j and a node in H ($e(j,k) = \emptyset$, if there is no edge between j and k). In line 4, l is set to the node that is not already in H and that has the largest number of edges connecting it to nodes already in H. This node is added to H in line 5 with all the edges connecting it to the nodes in H, so that H is always connected. The algorithm adds one node to H at each iteration of the while loop starting on line 2, until there are M nodes in H.

ACQA is based on SANP [8], which is an algorithm that can find end-to-end paths satisfying given QoS constraints in a graph where each node is an AS. In other words, SANP does not handle alliances. The basic idea of SANP and ACQA is the same: they both build a sub-graph by combining the neighborhoods of the nodes along the shortest path between the source and the destination.

V. Simulation Results

We have implemented ACQA in a simulator, in order to evaluate its performance and to compare it with SANP. The simulator takes as input a graph, the offers of each node and a list of connection requests. Each node in the graph is either an AS or an alliance. Each node is associated with a list of offers that specify the delay, the cost and the bandwidth available between any two of its entry points. In other words a node can be seen as a full mesh between all the ASBRs of the AS (or alliance). Both the cost and the delay of each offer are uniformly distributed on $[1, 50]$ and the capacity of each link is 1000. Connection requests arrive according to a Poisson process, with an average of one connection each 20 seconds. Each connection request specifies the source and destination AS and the constraints for the delay, the cost and the bandwidth. In order to simulate end-nodes, we introduce a "virtual" node (v_i) in each AS and in each alliance. We connect this node to all the ASBRs of its AS. The QoS parameters of these links are randomly selected for each connection, using the same distribution as the other intra-domain links.

In order to build an Internet-like AS-graph, we use Inet [13] to generate a graph with 2000 nodes, where each node is an AS. Then we use the MCL [14] algorithm to build nine alliances with 429, 188, 99, 65, 52, 42, 30, 28 and 25 nodes respectively. MCL uses a random walk to determine "attractive nodes" and "attracted nodes." An attractive node and its attracted nodes form an alliance. MCL ensures that a node cannot belong to different alliances and that each alliance is a connected graph. We use the original Inet graph with 2000 nodes to run SANP and the graph with the alliances to run ACQA. In both cases we set the neighborhood radius (r_0) to 1.

We have simulated 1000 requests using both SANP and ACQA. The source and destination AS are chosen at uniformly at random. The delay and cost constraints are uniformly distributed on $[1 \times 10^6, 10 \times 10^6]$ and the bandwidth constraint is 1. We have chosen these loose constraints in order to ensure that both SANP and ACQA will stop only when they have reached the destination and not because the constraints are not satisfied. (We have also considered the case where the

bandwidth is a limiting constraint and we present these results below.) For 22 requests all the paths found by SANP are dominated by *at least one* path found by ACQA; for these requests we can say that ACQA is better than SANP. For five requests we have the opposite. Note that in these cases, there is *at least one* solution in a set that dominates *all* the solutions in the other. Therefore in these cases it is possible to say that one algorithm is actually better than the other. For the remaining 973 requests we cannot immediately establish which algorithm is better. In the following section we present the metrics that we have used to characterize these solutions.

A. Comparing Solutions

Zitzler [10] introduced a metric that counts the average number of solutions in a set that are dominated by at least one solution in another set. Formally, for two sets of solutions A and B

$$C(A,B) \triangleq \frac{|\{b \in B; \exists a \in A : a \succeq b\}|}{|B|}$$

where $a \succeq b$ means that a dominates b and $|A|$ is the order (cardinality) of set A. $C(A,B) = 1$ means that each solution in B taken individually is dominated by at least one solution in A while $C(A,B) = 0$ means that none of the solutions in B are dominated by a solution in A. As an example, in Figure 5: $C(A,B) = \frac{1}{4}$ and $C(B,A) = \frac{0}{3} = 0$. Note that the situation corresponding to the 22 requests mentioned above *does not* correspond to the case the $C(A,B) = 1$; for example in the case of Figure 5 those 22 requests correspond to the case where there is *at least one* solution in A that dominates *all* the solutions in B.

Even though one could say that if path a dominates b ($a \succeq b$) then path a is better than path b, it is still worthwhile to compute the distance between these two paths. The generational distance G [9] computes how far a solution in a set is from its closest solution in the other set. We use a slightly modified version of G that computes the mean of the Euclidean distances between each solution in A and its closest solution in B. Formally,

$$G(A,B) \triangleq \frac{1}{n}\sum_{i=1}^{n} g_i.$$

where $n = |A|$. If a_i is dominated by its closest solution in B we set $g_i = -d_i$ where d_i is the euclidean distance between a_i and its closest solution in B, we set $g_i = d_i$ if a_i dominates its closest solution in B and $g_i = 0$ otherwise. As an example in Figure 5, $G(A,B) = \frac{g_1+g_2+g_3}{3}$ where g_1 is positive, g_2 and g_3 are equal to 0. If we compute $G(B,A)$, the distance between b_2 and its nearest solution in A (i.e., a_1) would be negative.

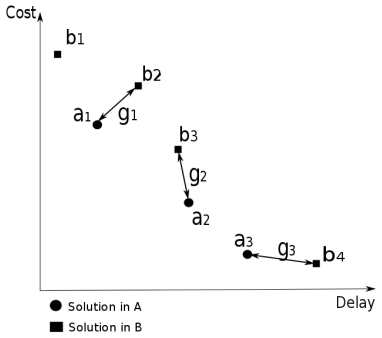

Fig. 5: Summary of the metrics

Using these metrics, we can now compare the 973 connections mentioned above. Figure 6 shows the CDF of $C(SANP, ACQA)$ and $C(ACQA, SANP)$. We remark that the CDF of $C(ACQA, SANP)$ is always below the CDF of $C(SANP, ACQA)$ meaning that the solutions found with ACQA tend to dominate more often the paths found with SANP than the opposite. Based on these observations we can conclude that for the 973 requests, the solutions found by ACQA are "better" than those found by SANP.

Figure 7 shows by "how much" the paths found by ACQA tend to dominate the paths found by SANP. For sake of clarity, when computing the CDF of $G(SANP, ACQA)$ and $G(ACQA, SANP)$, we do not take into account the cases where $G(SANP, ACQA) = 0$ and $G(ACQA, SANP) = 0$ respectively. $G(SANP, ACQA) = 0$ (resp $G(ACQA, SANP) = 0$) means that no solution found with SANP (resp ACQA) taken individually dominates or is dominated by its closest solution found with ACQA (resp SANP). As we are interested in the distance between two paths from different sets with one that dominates the other, we can skip this information. First, the CDF of $G(SANP, ACQA)$ is negative for 70% of the cases while $G(ACQA, SANP)$ is negative for 30% of the cases confirming that the paths found by ACQA tend to dominate the paths found by SANP. Second, given that the average cost (and delay) of a link is 25, by looking at the x-axis we can conclude that, when $G(SANP, ACQA)$ is negative (i.e., the solution found by ACQA dominates the solution found by SANP), the path found by ACQA is relatively better than the one found by SANP.

These results indicate that the "quality" of the paths found by ACQA is better than those found by SANP. Even though the constraints of each request are never a limiting factor, we can conclude that, if this were not the case, ACQA would be able to satisfy more connection requests than SANP given that it finds paths that tend to dominate those found by SANP. We have confirmed this

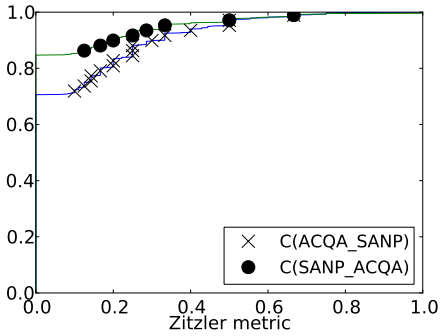

Fig. 6: Zitzler metric CDF

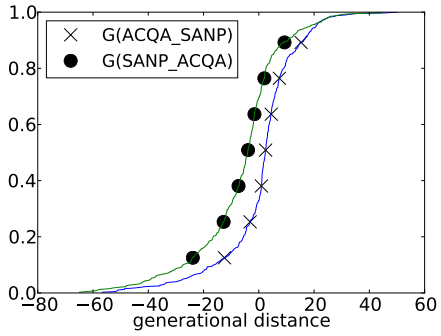

Fig. 7: Generational Distance CDF

by simulating 1000 requests with the same loose delay and cost constraints but with a capacity constraint of 999. As the capacity of each link is 1000, each link can support only one connection. Of these 1000 requests, 570 have been satisfied with SANP while 739 requests with ACQA. Once again ACQA outperforms SANP by satisfying more requests.

The improved performance of ACQA can be explained by the fact that some of the nodes in the sub-graph are alliances that can offer a greater diversity in terms of paths than single ASes. Yet this solution is still as scalable as SANP, given that the number of nodes in the sub-graph is of the same order of magnitude in ACQA and SANP. (Actually the sub-graph has fewer nodes in ACQA.)

VI. Conclusion

In this article we have presented ACQA, an algorithm capable of computing end-to-end QoS paths in a graph comprised of alliances and ASes, where both ASes and alliances publish a list of offers describing the QoS guarantees that they can offer between their entry and exit points (ASBRs). By constructing a sub-graph containing some of the nodes between the source and the destination, ACQA can compute a set of feasible non-dominated paths. The source can then select one of these paths based on its preferences.

We have shown that for the vast majority of the requests the paths found by ACQA are better than the paths found by SANP. Even though the size of the sub-graph is of the same order of magnitude. In the future, we want to investigate whether the way used to construct alliance-offers have an influence on the solutions found.

References

[1] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *Selected Areas in Communications, IEEE Journal on*, 1996.

[2] A. Lodhi, A. Dhamdhere, and C. Dovrolis, "Open peering by internet transit providers: Peer preference or peer pressure?" in *IEEE INFOCOM*, 2014.

[3] D. Barth, T. Mautor, and D. V. Monteiro, "Impact of alliances on end-to-end QoS satisfaction in an interdomain network," in *ICC*, 2009.

[4] N. Le Sauze, A. Chiosi, R. Douville, H. Pouyllau, H. Lonsethagen, P. Fantini, C. Palasciano, A. Cimmino, M. C. Rodriguez, and O. Dugeon, "ETICS: QoS-enabled interconnection for future internet services," *Future network and mobile summit*, 2010.

[5] N. Kumar and G. Saraph, "End-to-end QoS in interdomain routing," in *ICNS*, 2006.

[6] H. Pouyllau and R. Douville, "End-to-end QoS negotiation in network federations," in *NOMS Wksps, 2010 IEEE/IFIP*, 2010.

[7] H. Pouyllau and G. Carofiglio, "Inter-carrier SLA negotiation using q-learning," *Telecommunication Systems*, Jun. 2011.

[8] R. Jacquet, G. Texier, and A. Blanc, "SANP: an algorithm for selecting end-to-end paths with QoS guarantees," in *FutureNetworkSummit*, 2013.

[9] D. A. Van Veldhuizen, "Multiobjective evolutionary algorithms classifications, analyses, and new inovations," 1999.

[10] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions*, 1999.

[11] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. Van Der Merwe, "The case for separating routing from routers," in *ACM SIGCOMM*, 2004.

[12] J. Famaey, S. Latr, T. Wauters, and F. De Turck, "End-to-end resource management for federated delivery of multimedia services," *Journal of Network and Systems Management*, Sep. 2013. [Online]. Available: http://link.springer.com/10.1007/s10922-013-9288-y

[13] J. Winick and S. Jamin, "Inet-3.0: Internet topology generator," Technical Report CSE-TR-456-02, Tech. Rep., 2002.

[14] Stijn Van Dongen, "A cluster algorithm for graphs," 2000.