A Formal and Practical Framework for Constraint-Based Routing*†

Stefano Bistarelli
Dipartimento di Scienze
Università "G. d'Annunzio", Pescara, Italy
bista@sci.unich.it

Francesco Santini
IMT Institute for Advanced Studies
Lucca, Italy
f.santini@imtlucca.it

Abstract

We propose a formal model to represent and solve the Constraint-Based Routing *problem in networks. To attain this, we model the network adapting it to a weighted* or *graph (unicast delivery) or* and-or *graph (multicast delivery), where the weight on a connector corresponds to the cost of sending a packet on the network link modelled by that connector. We use the Soft Constraint Logic Programming (SCLP) framework as a convenient declarative programming environment in which to solve the routing problem. In particular, we show how the semantics of an SCLP program computes the best route in the corresponding graph. At last, we provide an implementation of the framework over scale-free networks.*

1 Introduction

Towards the second half of the nineties, Internet Engineering Task Force (IETF) and the research community have proposed many models and mechanisms to meet the demand for network *Quality of Service* (QoS). The classical routing problem has consequently been extended to include and to guarantee the QoS [16]: *QoS routing* [16, 9] denotes a class of routing algorithms that base path selection decisions on a set of QoS requirements or constraints, in addition to the destination. Service requirements have to be expressed in some measurable metric, such as bandwidth, number of hops, delay, jitter, cost and loss probability of packets. QoS-Routing and *Policy-Routing*, where the constraints are imposed by administrative policies (e.g. depending on traffic load, billing), are associated together under the name of *Constraint-Based Routing* (CBR) [16].

In this paper we propose a formal framework based on *Soft Constraint Logic Programming* (SCLP) [2, 3] in which it is possible to represent and solve QoS-Routing [5] (and CBR in general). First, we will describe how to represent a network configuration in a corresponding *or* graph (for the unicast delivery scheme) or *and-or* graph (for multicast), mapping network nodes to graph nodes and links to graph *connectors*. In the following, we will generally use the term *and-or* graph, or simply graph. QoS link costs will be translated into multidimensional costs for the associated connectors. Afterwards, we will propose the SCLP framework [2, 3] as a convenient declarative programming environment in which to specify and solve such problem. SCLP programs are an extension of usual Constraint Logic Programming (CLP) programs where logic programming is used in conjunction with soft constraints, that is, constraints which can be satisfied at a certain level. In particular, we will show how to represent an *and-or* graph as an SCLP program, and how the semantics of such a program computes the best route the corresponding weighted *and-or* graph (with route we will consider both multicast tree and unicast paths). SCLP is based on the general structure of *c-semiring* (or simply semiring), having the two operations $\times$ and +: the $\times$ is used to combine the costs, while the partial order defined by $+$ operation (see Sec. 3), is used to compare the costs. Notice that the cartesian product of two semirings is a semiring [4], and this can be fruitfully used to describe multi-criteria problems. In Sec. 6, we will suggest an implementation of the proposed framework to really test the performance on scale-free networks generated ad-hoc.

Related formal approaches, e.g. [11], adopt a hypergraph model in joint with semirings too, but the minimal path between two nodes is computed via a graphical calculous instead of SCLP. Moreover, as far as we know, none of the related works suggest a practical implementation of the presented formal ideas.

2 Constraint-Based Routing and Scale-free Networks

Constraint-Based Routing [16] refers to a class of routing algorithms that base path selection decisions on a set of requirements or constraints, in addition to destination cri-

*Supported by the MIUR PRIN 2005-015491.
†Partially supported by Institute for Informatics and Telematics (IIT-CNR) Pisa, Italy.

teria. These constraints may be imposed by QoS needs (i.e. QoS-Routing) or administrative policies (i.e. Policy-Routing), as already cited in Section 1. The aim of CBR is to reduce the manual configuration and intervention required for attaining traffic engineering objectives [14]; for this reason, CBR enhances the classical routing paradigm with special properties, such as being resource reservation-aware and demand-driven.

Policy-Routing selects paths that conform to administrative rules and *Service Level Agreements* (SLAs) stipulated among service providers and clients. For example, routing decisions can be based on the applications or protocols used, size of packets or identity of the communicating entities. Policy constraints can help improving the global security of the network and also help the resource allocation problem that includes business decisions. QoS routing instead attempts to simultaneously satisfy multiple QoS requirements requested by real-time applications: e.g. video conference, distributed simulation, stock quotes or multimedia entertainment.

Multiple metrics can certainly represent the requests more accurately than using a single measure. However, it is well known that the problem of finding a route subject to multiple constraints is inherently hard [16]. When some metrics take real or unbounded integer values [7], satisfying two boolean constraints (saying whether or not a route is feasible), or a boolean constraint and a quantitative constraint (i.e. optimizing a metric) is NP-complete [15, 16, 7]. For example the set of constraints $C = (delay \leq 40msec, \min(Cost))$ is intractable. For this reason, most of the implemented algorithms in this area apply heuristics to reduce the complexity. The unicast problem can be reconducted to the generic *Multi-Constrained Optimal Path* problem [7], while the multicast case refers to the *Constrained Steiner Tree* [16]; both these problems are NP-complete in their nature.

3 Soft Constraint Logic Programming

The SCLP framework [2, 3], is based on the notion of *c-semiring* introduced in [4]. A c-semiring S is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where A is a set with two special elements ($\mathbf{0}, \mathbf{1} \in A$) and with two operations $+$ and $\times$ that satisfy certain properties: $+$ is defined over (possibly infinite) sets of elements of A and thus is commutative, associative, idempotent, it is closed and $\mathbf{0}$ is its unit element and $\mathbf{1}$ is its absorbing element; $\times$ is closed, associative, commutative, distributes over $+$, $\mathbf{1}$ is its unit element, and $\mathbf{0}$ is its absorbing element (for the exhaustive definition, please refer to [4]). The $+$ operation defines a partial order $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$; we say that $a \leq_S b$ if b represents a value *better* than a. Other properties related to the two operations are that $+$ and $\times$ are monotone on $\leq_S$, $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum, $\langle A, \leq_S\rangle$ is a complete lattice and $+$ is its lub. Finally, if $\times$ is idempotent, then $+$ distributes over $\times$, $\langle A, \leq_S\rangle$ is a complete distributive lattice and $\times$ its glb.

Table 1. A simple SCLP program.

```
s(X)   :- p(X,Y).     q(a)   :- t(a).
p(a,b) :- q(a).       t(a)   :- 2.
p(a,c) :- r(a).       r(a)   :- 3.
```

Semiring-based constraint satisfaction problems (SCSPs) are constraint problems where each variable instantiation is associated to an element of a c-semiring A (to be interpreted as a cost, level of preference or, in this case, as a trust/reputation level), and constraints are combined via the $\times$ operation and compared via the $\leq_S$ ordering. Varying the set A and the meaning of the $+$ and $\times$ operations, we can represent many different kinds of problems, having features like fuzziness, probability, and optimization.

A simple example of a SCLP program over the semiring $\langle N, min, +, +\infty, 0\rangle$, where N is the set of non-negative integers and $D = \{a, b, c\}$, is represented in Tab. 1. The intuitive meaning of a semiring value like 3 associated to the atom $r(a)$ (in Tab. 1) is that $r(a)$ costs 3 units. Thus the set N contains all possible costs, and the choice of the two operations min and $+$ implies that we intend to minimize the sum of the costs. This gives us the possibility to select the atom instantiation which gives the minimum cost overall. Given a goal like $s(x)$ to this program, the operational semantics collects both a substitution for x (in this case, $x = a$) and also a semiring value (in this case, 2) which represents the minimum cost among the costs for all derivations for $s(x)$. To find one of these solutions, it starts from the goal and uses the clauses as usual in logic programming, except that at each step two items are accumulated and combined with the current state: a substitution and a semiring value (both provided by the used clause). The combination of these two items with what is contained in the current goal is done via the usual combination of substitutions (for the substitution part) and via the multiplicative operation of the semiring (for the semiring value part), which in this example is the arithmetic $+$. Thus, in the example of goal $s(X)$, we get two possible solutions, both with substitution $X = a$ but with two different semiring values: 2 and 3. Then, the combination of such two solutions via the min operation give us the semiring value 2.

4 Using *and-or* Graphs to Represent Networks with QoS Requirements

An *and-or* graph [12] is defined essentially as a hypergraph. Namely, instead of arcs connecting pairs of nodes

there are hyperarcs connecting an n-tuple of nodes ($n = 1, 2, 3, \ldots$). The arcs are called *connectors* and they must be considered as directed from their first node to all others. Formally an *and-or* graph is a pair $G = (N, C)$, where N is a set of *nodes* and C is a set of connectors $C \subseteq N \times \bigcup_{i=0}^{k} N^i$. Note that the definition allows 0-connectors, i.e. connectors with one input and no output node. In the following of the explanation we will also use the concept of *and* tree [12]: given an *and-or* graph G, an *and* tree H is a *solution tree of G with start node n_r*, if there is a function g mapping nodes of H into nodes of G such that: *i)* the root of H is mapped in n_r, and *ii)* if $(n_{i_0}, n_{i_1}, \ldots, n_{i_k})$ is a connector of H, then $(g(n_{i_0}), g(n_{i_1}), \ldots, g(n_{i_k}))$ is a connector of G.

Informally, a solution tree of an *and-or* graph is analogous to a path of an ordinary graph: it can be obtained by selecting exactly one outgoing connector for each node, and we use the resulting tree to model the multicast delivery. The unicast case is even simpler: we use an *or* graph (i.e. a classical graph) to represent the network and selecting one connector for each node clearly results in a path (not a tree).

In Fig. 1 we directly represent a very simple network as a weighted *and-or* graph. Each of the nodes can be easily cast in a corresponding node of the *and-or* graph. In Fig. 1, different icons feature the different role of the node in the network: the source of packets n_0, the routers n_1, n_2 and n_3, a subnetwork n_5 or plain receiver host n_4). To model the networks links between two nodes we use 1-connectors: (n_0, n_1), (n_1, n_2), (n_1, n_3), (n_2, n_4), (n_3, n_4) and (n_3, n_5). We remind that the connectors are directed, and thus, for example the connector (n_0, n_1) means that n_0 can send packets to n_1. Moreover, since we are possibly interested in a multicast communication, we need to represent the event of sending the same packet to multiple destinations at the same time. To attain this, in Fig. 1 we can see the two 2-connectors (n_1, n_2, n_3) and (n_3, n_4, n_5): we draw these n-connectors (with $n > 1$) as curved oriented arcs where the set of their output nodes corresponds to the destination nodes of the 1-connectors traversed by the curved arc. Considering the ordering of the nodes in the tuple describing the connector, the input node is at the first position and the output nodes (when more than one) follow the orientation of the related arc in the graph (in Fig. 1 this orientation is lexicographic). Notice that in the example we decided to use connectors with dimension at most equal to 2 (i.e. 2-connectors) for sake of simplicity. However it is possible to represent whatever cardinality (e.g. n) of multicast destination nodes (i.e. with a n-connector). 0-connectors are represented as a line ending with a square in Fig. 1 and are added only for receiver nodes.

In the example we propose here, we are interested in QoS link-state information concerning only the bandwidth and a generic money cost (e.g. to supply the service or to maintain a device). Bandwidth and cost can be seen as either QoS or policy constraints. Therefore, each link cost of the network can be labeled with a 2-dimensional cost for the related connector. For example, the pair $\langle 8, 1\rangle$ for the connector (n_0, n_1) tells us that the maximum bandwidth on that represented link is 80Mbps and a cost of 10€. In general, we could have a cost expressed with a v-dimensional vector, where v is the number of metrics to be taken in account while computing the best distribution tree. In the case when a connector represent a multicast delivery (i.e. a n-connector with $n > 1$), its cost is decided by assembling the costs of all the n links with the composition operation $\circ$, which takes as many v-dimensional cost vectors as operands, as the n number of links represented by the connector. For this example, the result of $\circ$ is the minimum bandwidth and the highest cost, ergo, the worst QoS metric values among the considered links:

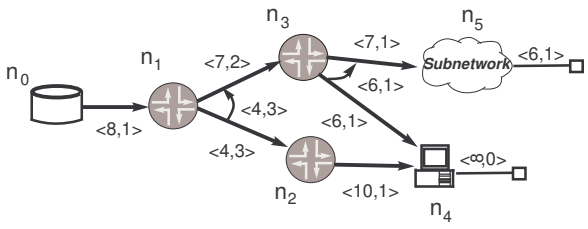

Figure 1. A network in *and-or* graph representation.

$$\circ(\langle b_1, c_1\rangle, \langle b_2, c_2\rangle, \ldots, \langle b_n, c_n\rangle) \longrightarrow$$

$$\langle \min(b_1, b_2, \ldots, b_n), \max(c_1, c_2, \ldots, c_n)\rangle$$

For example, the cost of the connector (n_1, n_2, n_3) in Fig. 1 is $\langle 4, 3\rangle$, since the costs of connectors (n_1, n_2) and (n_1, n_3) are respectively $\langle 4, 3\rangle$ and $\langle 7, 2\rangle$: $\circ(\langle 4, 3\rangle, \langle 7, 2\rangle) = \langle 4, 3\rangle$. All the costs of the connectors are reported in Tab. 2.

Then, we need some algebraic framework to model our preferences for the links in order to find the best route; to attain this, we use the semiring structure as described in Sec. 3. Since we are interested in maximizing the bandwidth of the distribution tree, we can use the c-semiring $S_{Bandwidth} = \langle \mathcal{R}^+, \max, \min, 0, +\infty\rangle$ (otherwise, we could be interested in finding the route with the minimal feasible bandwidth with $\langle \mathcal{R}^+, \min, \min, +\infty, 0\rangle$, for traffic engineering reasons). We can use $S_{Cost} = \langle \mathcal{R}^+, \min, +, +\infty, 0\rangle$ as the semiring to represent the cost, if we need to minimize it (here, $+$ is the arithmetic operator). Since the composition of c-semirings is still a c-semiring [4], $S_{Network} = \langle\langle \mathcal{R}^+, \mathcal{R}^+\rangle, +', \times', \langle 0, +\infty\rangle, \langle +\infty, 0\rangle\rangle$ is the adopted semiring, where $+'$ and $\times'$ correspond to the

vectorization of the $+$ and $\times$ operations in the two c-semirings: $\langle b_1, c_1\rangle +' \langle b_2, c_2\rangle = \langle \max(b_1, b_2), \min(c_1, c_2)\rangle$ and $\langle b_1, c_1\rangle \times' \langle b_2, c_2\rangle = \langle \min(b_1, b_2), c_1 + c_2\rangle$.

Clearly, the problem of finding best route is multi-criteria, since both bandwidth and delay must be optimized. We consider the criteria as independent among them, otherwise they can be rephrased to a single criteria [15]. Thus, the multidimensional costs of the connectors are not elements of a totally ordered set, and it may be possible to obtain several routes for the same destination (or destinations, if looking for a multicast distribution), all of which are not *dominated* by others, but which have different incomparable costs. The set of constraints for our problem is $C = (\max(Bandwidth), \min(Cost))$, which are both quantitative constraints: the semiring structure is suitable for metric optimization (i.e. to represent quantitative constraints), but in Sec. 5 we will apply also boolean constraints, e.g. only paths with $Cost < 22$€.

For each possible receiver node, the cost of its outgoing 0-connector will be always included in every route reaching it. As a remind, a 0-connector has only one input node but no destination nodes. If we consider a receiver as a plain node (e.g. n_4 in Fig. 1), we can set this cost as the $\mathbf{1}$ element of the adopted c-semiring ($\mathbf{1}$ is the unit element for $\times$), since the cost to reach the node is already completely described by the other connectors in the route: practically, we associate the highest possible QoS values to this 0-connector, in this case infinite bandwidth and null cost. Otherwise we can imagine a receiver as a more complex subnetwork (as n_5 in Fig. 1), and thus we can set the cost of the 0-connector as the cost needed to finally reach a node in that subnetwork (as the cost $\langle 6, 1\rangle$ for the 0-connector after node n_5 in Fig. 1), in case we do not want, or cannot, show the topology of the subnetwork, e.g. for security reasons.

5 *And-or* graphs using SCLP

To represent the network edges (i.e. 1-connectors), in SCLP we can write clauses like $edge(n_1, n_2) : -\langle 4, 3\rangle$, stating that the graph has a connector from n_1 to nodes n_2 and n_3 with a bandwidth cost of 40Mbps and a money cost of 30€. Other SCLP clauses can properly describe the structure of the route we desire to search over the graph.

We chose to represent an *and-or* graph with a program in *CIAO Prolog* [6], a system that offers a complete Prolog system supporting ISO-Prolog and several extensions. CIAO Prolog has also a fuzzy extension, but since it does not completely conform to the semantic of SCLP defined in [3] (due to interpolation in the interval of the fuzzy set), we decided to use the CIAO operators among constraints (as $<$ and $\leq$), and to model the $\times$ operator of the c-semiring with them. For this reason, we added the cost of the connector in the head of the clauses, differently from SCLP clauses

Table 2. The CIAO program representing all the routes over the weighted *and-or* graph problem in Fig. 1.

```
:- module(network,_,_).
:- use_module(library(lists)).

min([X, Y], X) :- X < Y.
min([X, Y], Y) :- X >= Y.
max([X, Y], X) :- X > Y.
max([X, Y], Y) :- X =< Y.

times([B1, C1], [B2, C2], [B, C]) :-
    min([B1, B2], B),
    C is (C1 + C2).

Leaves
    leaf([n4], [1000, 0]).
    leaf([n5], [6, 1]).

Edges
    edge(n0, [n1], [8, 1]).
    edge(n1, [n2], [4, 3]).
    edge(n1, [n3], [7, 2]).
    edge(n2, [n4], [10, 1]).
    edge(n3, [n4], [6, 1]).
    edge(n3, [n5], [7, 1]).

5)  routeList([X|Xs], Z, [B, C]):-
      route(X, Z1, [B1, C1]),
      append(Z1, Z2, Z),
      routeList(Xs, Z2, [B2, C2]),
      times([B1, C1], [B2, C2], [B, C]).

1)  connector(X, [Y], L, [B,C]):-
      nocontains(L, Y),
      edge(X, Y, [B,C]).

    connector(X, [Y|Ys], L, [B,C]):-
      edge(X, Y, [B1,C2]),
      nocontains(L,Y),
      insert_last(L, Y, Z),
      connector(X, Ys, Z, [B2,C2]),
      min([B1,B2], B]),
      max([C1,C2], C).

2)  route(X, [X], [B, C]):-
      leaf([X], [B, C]).

3)  route(X, Z, [B, C]):-
      connector(X, W, [], [B1, C1]),
      routeList(W, Z, [B2, C2]),
      times([B1, C1], [B2, C2], [B, C]).

4)  routeList([], [], [100, 0]).
```

which have the cost in the body of the clause.

From the weighted *and-or* graph problem in Fig. 1 we can build the corresponding CIAO program of Tab. 2 as follows. The set of network edges (or 1-connectors) is highlighted as *Edges* in Tab. 2. Each fact has the structure

$$edge(source\_node, [dest\_nodes], [bandwidth, cost])$$

e.g. the fact $edge(n_1, [n_2], [4, 3])$ represents the 1-connector of the graph (n_1, n_2) with bandwidth equal to 40Mbps and cost 30€. The *Rules 1* in Tab. 2 are used to compose the edges (i.e. the 1-connectors) together in order to find all the possible n-connectors with $n \geq 1$, by aggregating the costs of 1-connectors with the $\circ$ composition operator, as described in Sec. 4 (the lowest of the bandwidths and the greatest of the costs of the composed 1-connectors). Therefore, with these clauses (in *Rules 1*) we can automatically generate the set of all the connectors outgoing from the considered node (in Table 2, *nocontains* and *insert_last* are CIAO predicates used to build a well-formed connector). The *Leaves* in Table 2 represent the 0-connectors (a value of 1000 represents ∞ for bandwidth). The *time* rule in Table 2 mimics the $\times$ operation of the semiring proposed in Section 4: $S_{Network} = \langle\langle \mathcal{R}^+, \mathcal{R}^+\rangle, +', \times', \langle 0, +\infty\rangle, \langle +\infty, 0\rangle\rangle$, where $+'$ is equal to $\langle \max, \min\rangle$ and $\times'$ is equal to $\langle \min, +\rangle$, as defined in Section 4. At last, the rules 2-3-4-5 of Table 2 describe the structure of the routes we want to find over the graph. *Rule 2* represents a route made of only one leaf node, *Rule 3* outlines a route made of a connector plus a list of sub-routes with root nodes in the list of the destination nodes of the

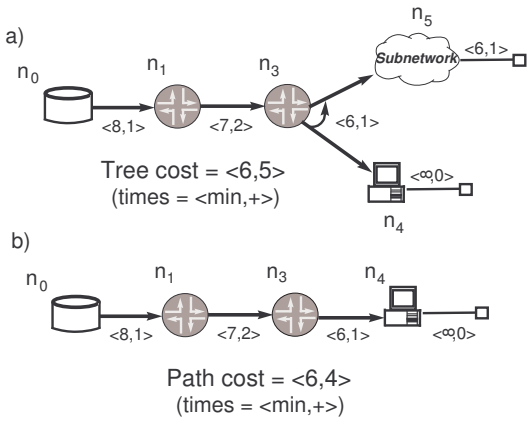

Figure 2. *a)* **The best multicast tree among** n_0 **and** n_4**-**n_5**, and** *b)* **the best unicast path between** n_0 **and** n_4**.**

connector, *Rule 4* is the termination for *Rule 5*, and *Rule 4* is needed to manage the junction of the disjoint sub-routes with roots in the list $[X|Xs]$; clearly, when the list $[X|Xs]$ of destination nodes contains more than one node, it means we are looking for a multicast route. When we compose connectors or trees (*Rule 2* and *Rule 5*), we use the *times* rule to compose their costs together. In *Rule 5*, *append* is a CIAO predicate used to join together the lists of destination nodes, when the query asks for a multicast route.

To solve the CBR problem it is enough to perform a query in the Prolog language: for example, if we want to compute the cost of all the multicast trees rooted at n_0 and having as leaves the nodes representing the receivers (in this case, n_4 and n_5), we have to perform the query $route(n_0, [n_4, n_5], [B, C])$, where B and C variables will be instantiated with the bandwidth and cost of the found trees. For this query, the best output (in terms of the adopted QoS metrics) of the CIAO program corresponds to the cost of the tree in Fig. 2*a*, i.e. $\langle 6, 5\rangle$, since $\times'$ computes the *minimum bandwidth - cost sum* of the connectors.

The best unicast path between n_0 and n_4 can instead be found with the query $route(n_0, [n_4], [B, C])$, and it is represented in Fig. 2*b*; its cost is $\langle 6, 4\rangle$. Notice that the best path or tree is directly computed by the SCLP engine as described in the example in Sec. 3: given a query, the operational semantics collects a semiring value which represents the best cost (w.r.t. the $+$ operator) among the costs of all the derivations satisfying the query. In Tab. 2, the SCLP engine is prototyped with a CIAO Prolog program.

As anticipated in Sec. 4, semiring structures are the ideal to represent quantitative constraints since the $+$ operation of the semiring defines a partial order over A (see Sec. 3), i.e. over the set of QoS metric values. This operation can be consequently used to optimize the route. However, also boolean constraints, e.g. a route is accepted only if its cost is below a given threshold (e.g. $Cost < 30$€), can be modeled in our framework. For example, with the query $route(n_0, [n_4], [B, C]), C < 3$ no path is returned since the best possible path in Fig. 2 has a money cost equal to 4. The $C < 3$ requirement can be directly embedded in the *times* rule of the CIAO program Tab. 2, in order to also optimize the search by stopping it as soon as $C < 3$ is no longer true.

6 Implementing the Framework

To develop and test a practical implementation of our model, we adopt the *Java Universal Network/Graph Framework* (JUNG) [13], a software library for the modeling, analysis, and visualization of a graph or network. With this library it is also possible to generate scale-free networks according to the preferential attachment proposed in [1]: each time a new vertex v_n is added to the network G, the probability p of creating an edge between an existing vertex v and v_n is $p = (degree(v) + 1)/(|E| + |V|)$, where $|E|$ and $|V|$ are respectively the current number of edges and vertices in G. Therefore, vertices with higher degree have a higher probability of being selected for attachment. We generated the scale-free network and then we automatically produced the corresponding program in CIAO, as in Sec. 5. The reported statistics suggest the scale-free nature of our network: 265 nodes, 600 edges, clustering coefficient $= 0.13$, Average Shortest Path $= 3.74$, Min/Max/Avg Degree $= 1/20/4.52$ and Diameter $= 8$. Therefore, a quite high clustering coefficient, a low average shortest path and a high variability of vertex degrees (between average and max). These features are evidences of the presence of few big hubs that can be used to shortly reach the destinations.

Together with the bandwidth/cost quantitative constraint of Sec. 5, we added the $Path\_Hops < 2 \cdot Avg\_Shortest\_Path$ boolean constraint: in this case, $Path\_Hops < 8$, which is also the diameter of the network. This constraint limits the search space and provides a good approximation at the same time: in scale-free networks, the average distance between two nodes can be $\ln \ln N$, where N is the number of nodes [8]. Therefore, this hop constraint can be successfully used also with large networks, and limiting the depth to twice the average shortest path value still results in a large number of alternative routes. Other boolean constraints can be used to further prune the search as soon as they are violated, consequently improving the performance. We performed 50 queries for the unicast case (i.e. between two random nodes) and all the results were found within 5 minutes. Concerning instead quantitative constraints, an already obtained path p can be used to immediately discard successive partial results during the search, if the current QoS values are already worse than p

(i.e. we can use the QoS features of p as a "cut" level). Unfortunately, it is not possible to produce an uniform performance graph by considering different node numbers for the generated graphs, since, using the same input parameters, JUNG generates a lot of disconnected networks when setting a lower number of desired final nodes.

If we consider the multicast case, each of the nodes in the *and-or* graph has $O(2^d)$ connectors, where d is the degree of the node. This exponential result is not a problem for most of nodes, since in scale-free networks the average degree is very low. The problem arises for the few hub nodes; in this case we can reduce the n-connector facts basing on the number of destinations: if only 5 destinations must be reached, it is useless to create 6-connectors or more. Moreover, we can use a variable in the clauses to count the current width of the partial tree solution in order to prevent the search from widening more than the number of destinations.

We think there are several way to further improve the computational results. The complexity of the tree search can be reduced by using *tabling* (or *memoing*) techniques [10]. The calls to *tabled* predicates are stored in a searchable structure together with their proven instances, and subsequent identical calls can use the stored answers without repeating the computation. One more improvement is the inclusions of facts describing the topology of the network (or part of it). In this way, like in ordinary network routing, we can immediately remove from the search the not involved clusters or the clusters we do not want to cross.These improvements are strongly needed for hub nodes, i.e. the backbone nodes of the network , which represent however a small percentage of the total number of nodes.

7 Conclusions

We have described a method to represent and solve the CBR problem with the combination of *and-or* graph and the declarative SCLP environment: the best multicast or unicast route found on an *and-or* graph corresponds to the semantics of a SCLP program. The route satisfies multiple constraints regarding QoS requirements, e.g. minimizing the global bandwidth consumption, reducing the delay, or accepting only the routes that use k hops at most. The semiring structure is a very parametric tool where to represent different QoS metrics. Since it is well-known that even a shortest path problem with two or more independent metrics is NP-complete (see Sec. 1), we have proposed a framework based on AI techniques (i.e. soft constraints). The convenience is to use a declarative framework where constraints on the routes can be easily represented. Moreover we have provided a practical implementation of the framework and a test on a scale-free network, whose results are quite promising. The framework can be used to prototype and test new constraints in small networks (i.e. 100-1000 nodes) or parts of wider graphs.

Concerning future works, we want to produce more tests, also with different scale-free/small-world topology generators. We plan to improve the computational results by adding to the program some clauses that describe the topology of the network. Moreover, we will study ad-hoc memoization techniques to reduce the complexity of big hubs.

References

[1] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

[2] S. Bistarelli. *Semirings for Soft Constraint Solving and Programming*, volume 2962 of *Lecture Notes in Computer Science*. Springer, 2004.

[3] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint logic programming. In *Proc. IJCAI97 (Morgan Kaufman)*, pages 352–357. Morgan Kaufman, 1997.

[4] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint solving and optimization. *Journal of the ACM*, 44(2):201–236, 1997.

[5] S. Bistarelli, U. Montanari, F. Rossi, and F. Santini. Modelling multicast qos routing by using best-tree search in and-or graphs and soft constraint logic programming. *Electr. Notes Theor. Comput. Sci.*, 190(3):111–127, 2007.

[6] F. Bueno, D. Cabeza, M. Carro, M. Hermenegildo, P. López-García, and G. Puebla. The ciao prolog system: reference manual. Technical Report CLIP3/97.1, School of Computer Science, Technical University of Madrid (UPM), 1997.

[7] S. Chen and K. Nahrstedt. An overview of quality of service routing for next-generation high-speed networks: Problems and solutions. *IEEE Network*, 12(6):64–79, 1998.

[8] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *Phys. Rev. Lett.*, 90(5):058701, Feb 2003.

[9] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. RFC 2386: A framework for QoS-based routing in the Internet, August 1998. Informational.

[10] B. Cui and D. S. Warren. A system for tabled constraint logic programming. In *CL '00: Conference on Computational Logic*, pages 478–492. Springer-Verlag, 2000.

[11] D. Hirsch and E. Tuosto. SHReQ: coordinating application level QoS. In *SEFM '05: Software Engineering and Formal Methods*, pages 425–434. IEEE Computer Society, 2005.

[12] A. Martelli and U. Montanari. Optimizing decision trees through heuristically guided search. *Commun. ACM*, 21(12):1025–1039, 1978.

[13] J. O'Madadhain, D. Fisher, S. White, and Y. Boey. The JUNG (Java Universal Network/Graph) framework. Technical report, UC Irvine, 2003.

[14] E. Rosen, A. Viswanathan, and R. Callon. IETF-RFC3031: Multiprotocol Label Switching Architecture, 2001.

[15] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, 1996.

[16] O. Younis and S. Fahmy. Constraint-based routing in the internet: basic principles and recent research. *IEEE Communications Surveys and Tutorials*, 5(1):2–13, 2003.