Modelling Multicast QoS Routing by using Best-Tree Search in AND-OR Graphs and Soft Constraint Logic Programming

Stefano Bistarelli[1], Ugo Montanari[2], Francesca Rossi[3] and Francesco Santini[4]

[1] Dipartimento di Scienze, Università G. D'Annunzio" di Chieti-Pescara, Viale Pindaro 87, 65127 Pescara (Italy), e-mail: bista@sci.unich.it *
[2] Dipartimento di Informatica, Università di Pisa, Largo Bruno Pontecorvo 3, 56127 Pisa (Italy), e-mail: ugo@di.unipi.it.
[3] Dipartimento di Matematica Pura ed Applicata, Università di Padova, Via Belzoni 7, 35131 Padova (Italy), e-mail: frossi@math.unipd.it
[4] IMT Lucca Institute for Advanced Studies, Via S. Micheletto 3, 55100 Lucca (Italy), e-mail: francesco.santini@imtlucca.it *

Abstract. We suggest a formal model to represent and solve the multicast routing problem in multicast networks. To attain this, we draw the network adapting it to a weighted AND-OR graph, where the weight on a connector corresponds to the cost of sending a packet on the network link modelled by that connector. Then, we use the Soft Constraint Logic Programming (SCLP) framework as a convenient declarative programming environment where to specify related problems. In particular, we show how the semantic of a SCLP program computes the best tree in the corresponding AND-OR graph: this result can be adopted to find, from a given source node, the multicast distribution tree having minimum cost and reaching all the destination nodes of the multicast communication. The costs on the connectors can be described also as vectors (multidimensional costs), each component representing a different *Quality of Service* metric value. Therefore, the construction of the best tree may involve a set of criteria to be all optimized (*multi-criteria* problem), e.g. maximum global bandwidth and minimum delay that can be experienced on a single link.

1 Motivation and main idea

Multicast is an important bandwidth-conserving technology that reduces traffic by simultaneously delivering a single stream of information to multiple receivers. Therefore, while saving resources, multicast is well suited to concurrently distribute contents on behalf of applications asking for a certain timeliness of delivery: thus, multicast routing has naturally been extended to guarantee *Quality of Service* (QoS) requirements [19].

In this paper we suggest a formal model to represent and solve the multicast routing problem in multicast networks with QoS. For the representation we use AND-OR graphs [13] to model the network and SCLP programs [1, 4, 9] on the graphs to compute the best tree according to QoS criteria.

* Partially supported by Institute of Informatics and Telematics (IIT-CNR) Pisa, Italy

Given a multicast group of receiver nodes and a set of optimization objective functions, the multicast routing is the process of building a multicast tree that optimize these functions, often expressing the aim of minimizing the cost of the tree. If we are dealing with QoS requirements, a set of constraints is also given: constraints are in the form of, for example, end-to-end delay bounds, jitter bound, minimum bandwidth of the path or other QoS metrics. The resulting multicast tree must provide both reachability from source to all destinations, and the QoS constraints.

First, we will describe how to represent a network configuration in its corresponding AND-OR graph, mapping network nodes to AND-OR graph nodes and links to graph *connectors*. QoS link costs will be translated in costs for the connectors. Generally, AND/OR graphs are used to model problem solving processes, and together with minimum cost solution graphs have been studied extensively in artificial intelligence [16].

Afterwards, we will propose the Soft Constraint Logic Programming (SCLP) framework [1, 4, 9] as a convenient declarative programming environment where to specify and solve such problem. SCLP programs are an extension of usual Constraint Logic Programming (CLP) [10] 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 tree in the corresponding weighted AND-OR graph. Therefore, the best tree found in this way, can be used to shape the optimized multicast tree that ensures QoS requirements on the corresponding network.

Since several QoS parameters express the cost of a link at the same time, this problem can be addressed as a *multi-criteria* problem [6], where the combination of the costs is done via an operator which is more general than the usual sum of the link weights. This extension can be easily cast within the SCLP programming framework, because it is based on the general structure of a *semiring* with two operations ($\times$ and +). Then, $\times$ is used to combine the costs, while $+$ and the implied partial order, to compare them.

The work presented and suggested in this paper extends some results on shortest path problems presented in [5] and [6] . The main idea underlying this extension concerns the use of non-linear clauses in SCLP, that is, clauses which have more than one atom in their body: in this way, we can represent trees instead of paths.

This paper is organized as follows: in Sec. 2 we present some general background information about multicast routing, including also its QoS extensions, then in Sec. 3 we describe the problem of finding the best weighted tree in an AND-OR graph. Section 4 features the SCLP framework, while Sec. 5 depicts how to represent a network environment with an AND-OR graph. At last, in Sec. 6 we describe the way to pass from and-or graph to SCLP programs, showing that the semantic of SCLP program is able to compute the best tree in the corresponding AND-OR graph. This tree represents the solution: the multicast tree that optimizes QoS requirements. Section 7 draws the final conclusions and outlines intentions for future works.

2 Multicast Routing with QoS extensions

Given a node generating packets, we can classify network data delivery schemas into three main types: *i) Unicast*, when data is delivered from one sender to one specific recipient, providing one-to-one delivery, *ii) Broadcast*, when data is instead delivered to all hosts, providing one-to-all delivery, and finally, *iii) Multicast*, when data is delivered to all the selected hosts that have expressed interest; thus, this last method provides one-to-many delivery.

In this paper we concentrate on the third paradigm, since our intention is to provide a solution to the problem of transmitting a data packet from one source to K receivers. In its simplest implementation, multicast can be provided using multiple unicast transmissions, but with this solution, the same packet can traverse the same link multiple times. For this reason, the network must provide this service natively.

A multicast address is also called a multicast group address, with which the routers can locate and send packets to all the members in the group. A group member is a host that expresses interest in receiving packets sent to a specific group address. A group member is also sometimes called a *receiver* or a *listener*. A multicast source is a host that sends packets with the destination address set to a multicast group. To deliver data only to interested parties, routers in the network build a *multicast* (or *distribution*) *tree* (Figure 1). Each subnetwork that contains at least one interested listener is a leaf on the tree. Where the tree branches, routers replicate the data and send a single packet down each branch. No link ever carries a duplicate flow of packets, since packets are replicated in the network only at the point where paths diverge, reducing the global traffic. *Downstream* is in the data flow direction toward the receivers, while *Upstream* is in the direction toward the source.

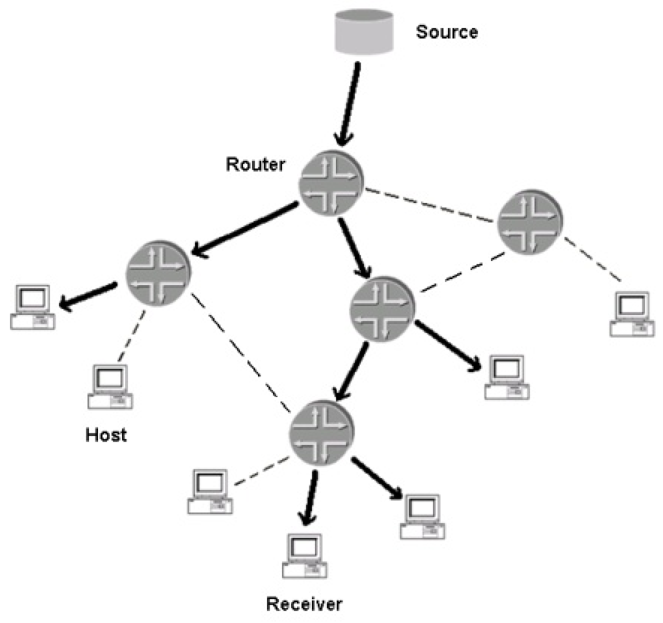

Fig. 1: An example of a multicast distribution tree built on a network: oriented arcs highlight the tree (direction is down stream), while dashed lines correspond to links not traversed by the flow.

Applications that take advantage of multicast routing include, for example, video conference, corporate communications, distance learning, distributed simulation, resource discovery, software distribution, stock quotes, news and also entertainment applications such as, video-on-demand, games, interactive chatlines and internet jukebox.

Since many applications that need multicast distribution also require a certain timeliness of delivery (*real-time* applications), multicast routing has been clearly extended to include and guarantee QoS requirements; a global picture of QoS is given in [21]. In this case, the *Constraint-Based* multicast routing, the problem is to find the best distribution tree with respect to certain performance related constraints, to better utilize network resources and to support QoS requirements of the applications. Constraint-Based Routing (CBR) denotes a class of routing algorithms that base path selection decisions on a set of requirements or constraints, in addition to the destination: constraints can be imposed by administrative policies, or by QoS needs [22]. The other intent of CBR is to increase the utilization of the network (CBR is a tool for *Traffic Engineering* [21, 22]), and is a part of the global framework that provide Internet QoS [21].

Multicast problem has been studied with several algorithms and variants, such as *Shortest-Path Tree* (SPT), *Minimum Spanning Tree* (MST), *Steiner Tree* (ST), *Constrained Steiner Tree* (CST), and other miscellaneous trees [19]. Algorithms based on SPT (e.g. Dijkstra or Bellman-Ford [8]) aim to minimize the sum of the weights on the links from the source to each receiver, and if all the link cost one unit, the resulting tree is the least-hop one. The MST (e.g. Prim algorithm [8]) span all the receivers in the multicast group, minimizing the total weight of the tree at the same time; at each step, the tree is augmented with an edge that contributes the minimum amount possible to the total cost of the tree, so the algorithm is greedy. A ST [20] is a tree which spans a given subset of vertices in the graph with the minimal total distance on its edges. If the subset matches the entire multicast group, ST problem reduces to MST. ST has been extended to CST, including side constraints concerning QoS metrics. ST and CST are NP-Complete problems [20], and many heuristics have been proposed to efficiently deal with them [19, 20].

The most popular solutions to multicast routing involve tree construction. There are two reasons for basing efficient multicast routes on trees: *i)* the data can be transmitted in parallel to the various destinations along the branches of the tree, and *ii)* a minimum number of copies of the data are transmitted.

Multicast QoS routing is generally more complex than unicast QoS routing, and less proposals have been elaborated in this area [22]. With respect to unicast, the additional complexity stems from the need to support shared and heterogeneous reservation styles (towards distinct group members) and global admission control of the distribution flow. Some of the approaches use a Steiner formulation [11] or extend existing algorithm to optimize the delay (MOSPF [15] is the multicast version of OSPF), while the *Delay Variation Multicast Algorithm* (DVMA) [18] computes a multicast tree with both bounded delay and bounded jitter. Also, delay-bounded and cost-optimized multicast routing can be formulated as a Steiner tree: an example approach is *QoS-aware Multicast Routing Protocol* [15] (QMRP). Other multicast QoS routing algorithms and related problems (entailing stability, robustness and scalability) are presented in [22], and we did not include them here for lack of space.

3 AND-OR Graphs and Best Solution Trees

An AND-OR graph [13] is defined essentially as a hypergraph. Namely, instead of arcs connecting pairs of nodes there are hyperarcs connecting n-tuple of nodes ($n = 1, 2, 3, \ldots$). Hyperarcs 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

$$C \subseteq N \times \bigcup_{i=0}^{k} N^i.$$

Each k-connector $(n_{i_0}, n_{i_1}, \ldots, n_{i_k})$ is an ordered $(k+1)$-tuple, where n_{i_0} is the *input* node and $n_{i_1}, \ldots, n_{i_k}$ are the *output* nodes. We say that n_{i_0} is the *predecessor* of $n_{i_1}, \ldots, n_{i_k}$ and these nodes are the *successors* of n_{i_0}. Note that when $C \subseteq N^2$ we have a usual graph whose arcs are the 1-connectors. Note that there are also 0-connectors, i.e., connectors with one input and no output node.

In Figure 2 we give an example of an AND-OR graph, whose nodes are $n_0, \ldots, n_8$. The 0-connectors are represented as a line ending with a square, whereas k-connectors ($k \geq 0$) are represented as k directed lines connected together. For instance, (n_0, n_1) and (n_0, n_5, n_4) are the 1-connector and 2-connector, respectively, with input node n_0.

An AND tree is a special case of an AND-OR graph, where every node appears exactly twice in the set of connectors C, once as an input node of some connector, and once as an output node of some other connector. The only exception is a node called *root* which appears only once, as an input node of some connector. The *leaves* of an AND tree are those nodes which are input nodes of a 0-connector. An example of an AND tree with root n'_2 is given in Figure 3. Here n'_7, n'_8 and n''_8 are leaves.

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:

– the root of H is mapped in n_r.
– 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. For instance, the AND tree in Fig. 3 is a solution tree of the graph in Fig. 2 with start node $n_r = n_2$, if we let $g(n'_2) = n_2, g(n'_4) = n_4, g(n'_5) = n_5, g(n'_7) = n_7, g(n'_8) = n_8$ and $g(n''_8) = n_8$. Note that distinct nodes of the tree can be mapped into the same node of the graph.

The AND-OR graph in Fig. 2 has a k-adic function over the reals (*cost function*) associated with each k-connector, and is therefore defined as *functionally weighted* AND-OR *graph*. In particular, a constant is associated with each 0-connector. It is easy to see that if the functionally weighted graph is an AND tree H, a *cost* can be given to it, just evaluating the functions associated with its connectors. Recursively, to every subtree of H with root in node n_{i_0} a cost c_{i_0} is given as follows:

– If n_{i_0} is a leaf, then its cost is the associated constant.

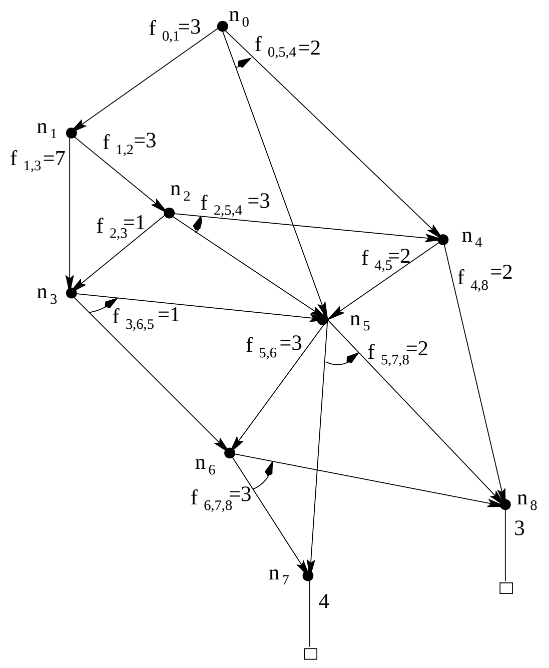

Fig. 2: A weighted AND-OR graph problem.

– If n_{i_0} is the input node of a connector $(n_{i_0}, n_{i_1}, \dots, n_{i_k})$, then its cost is $c_{i_0} = f_r(c_{i_1}, \dots, c_{i_k})$ where f_r is the function cost associated with the connector, and $c_{i_1}, \dots, c_{i_k}$ are the costs of the subtrees rooted at nodes $n_{i_1}, \dots, n_{i_k}$.

The general optimization problem can be stated as follows: *given a functionally weighted* AND-OR *graph, find a minimal cost solution tree with start node* n_r. The function used to assign a value to the input node n_{i_0} of a k-connector $(n_{i_0}, n_{i_1}, \dots, n_{i_k})$ is of the form $f_r(c_{i_1}, \dots, c_{i_k}) = a_r + c_{i_1} + \dots + c_{i_k}$ where a_r is a constant associated to the connector and $c_{i_1}, \dots, c_{i_k}$ are the costs of the subtrees rooted at nodes $n_{i_1}, \dots, n_{i_k}$. Therefore, the cost of the tree in Fig. 3, with root node n_2, is 17.

In the following of this paper we will show that this cost function is only an instantiation of a more general one based on the notion of *c-semiring* [1, 2].

4 Soft Constraint Logic Programming

The SCLP framework [1, 4, 9], is based on the notion of *c-semiring* introduced in [2, 3]. 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

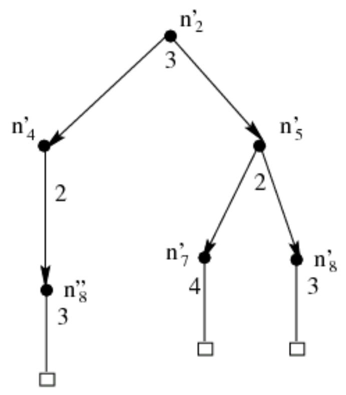

Fig. 3: A minimal cost solution AND tree for the graph in Fig. 2, with start node $n_r = n_2$.

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 [3]).

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.

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, ...), 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. Moreover, in [3] we have shown that the cartesian product of two c-semirings is another c-semiring, and this can be fruitfully used to describe multi-criteria constraint satisfaction and optimization problems.

The SCLP framework extends the classical constraint logic programming formalism [10] in order to be able to handle also SCSP [2, 3] problems. In passing from CLP to SCLP languages, we replace classical constraints with the more general SCSP constraints where we are able to assign a *level of preference* to each instantiated constraint (i.e. a ground atom). To do this, we also modify the notions of interpretation, model, model intersection, and others, since we have to take into account the semiring operations and not the usual CLP operations.

The fact that we have to combine several refutation paths when we have a partial order among the elements of the semiring (instead of a total one), can be fruitfully used in the context of this paper when we have an AND-OR graph problem with incomparable costs associated to the connectors. In fact, in the case of a partial order, the solution of the problem of finding a *best* tree should consists of all those trees whose cost is not "dominated" by others.

Table 1: A simple example of an SCLP program.

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

A simple example of an 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 Table 1. The choice of this semiring allows to represent constraint optimization problems where the semiring elements are the costs for the instantiated atoms. The intuitive meaning of a semiring value like 3 associated to the atom $r(a)$ 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 for 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 $+$. 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.

5 Using AND-OR Graphs to represent QoS multicast networks

In this Section we explain a method to traslate the representation of a multicast network with QoS requirements (Figure 5*a*) into a corresponding weighted AND-OR graph model (Figure 5*b*). This procedure can be split in three distinct steps, respectively focusing on the representation of *i)* network nodes, *ii)* network links and *iii)* link costs in terms of QoS metrics.

Each of the network nodes can be easily cast in the corresponding AND-OR graphs as a single graph node: thus, each node in the graph can represent an interconnecting device (e.g. a router), or a node acting as the source of a multicast communication (injecting packets in the network), or, finally, a receiver belonging to a multicast group and participating to the communication. In Sec. 6, when we will look for the best tree solution, the root of the best AND tree will be mapped in the node representing the source of the multicast communication; in the same way, receivers will be modelled by the leaves of the resulting AND tree. When we translate a receiver, we add an outgoing 0-connector (Figure 5*b*), whose meaning (cost) will be explained next in this Section. Suppose that $\{n_0, n_1, \dots, n_9\}$ in Figure 5*a* are the identifiers of the network nodes.

To model the links, we examine the forwarding star (*f-star*) of each node in the network: we consider the links as oriented, since the cost of sending packets from node n_i to n_j can be different from the cost of sending from n_j to n_i (one not oriented link can be easily replaced by two oriented ones). Supposing that the f-star of node n_i includes the arcs (n_i, n_j), (n_i, n_k) and (n_i, n_z), we translate this f-star by constructing one connector directed from n_i to each of the subsets of destination nodes $\{j, k, z\}$ (Fig. 4), for a total number of $2^n - 1$ subsets, excluding $\{\emptyset\}$. Thus, all the resulting connectors with n_i as the input node are (n_i, n_j), (n_i, n_k), (n_i, n_z), (n_i, n_k, n_j), (n_i, n_k, n_z), (n_i, n_j, n_z) and (n_i, n_j, n_k, n_z).

To simplify Fig. 4*b*, arcs linking directly two nodes represent 1-connectors (n_i, n_j), (n_i, n_k) and (n_i, n_z), while curved oriented lines represent n-connectors (with $n > 1$), where the set of their output nodes correspond to the output nodes of the traversed arcs. With respect to n_i, in Fig. 4 we have curved line labelled with *a* that correspond to (n_i, n_k, n_j, n_z), *b* to (n_i, n_k, n_j), *c* to (n_i, n_j, n_z), and, at last, *d* to (n_i, n_k, n_z). To have a clear figure, the network links in Fig. 5*a* are oriented "towards" the receivers, thus we put only the corresponding connectors in Fig 5*b*.

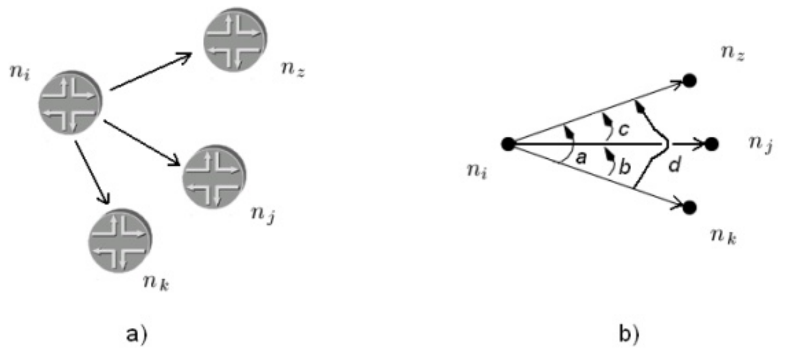

Fig. 4: *a)* the f-star of n_i network-node and *b)* its representation with connectors.

In the example we propose here, we are interested in QoS link-state information concerning only bandwidth and delay. Therefore, each link of the network can be labeled with a 2-dimensional cost, for example the pair $\langle 7, 3 \rangle$ tells us that the maximum bandwidth on that specific link is 70 Mbs and the maximum delay is 30 ms. In general, we could have a cost expressed with a n-dimensional vector, where n is the number of metrics to be taken in account while computing the best distribution tree. Since we want to maintain this link state information even in the AND-OR graph, we label the corresponding connector with the same tuple of values (Figure 5).

In case a connector represent more than one network link, its cost is decided by assembling the costs of the these links with the composition operation $\circ$, which takes as many n-dimensional vectors as operands, as the number of links represented by the connector. Naturally, we can instantiate this operation for the particular types of costs adopted to express QoS: for the example given in this Section, the result of $\circ$ is the minimum bandwidth and the highest delay, ergo, the worst QoS metric values:

$$\circ(\langle b_1, d_1\rangle, \langle b_2, d_2\rangle, \ldots, \langle b_n, d_n\rangle) \longrightarrow \langle \text{min}(b_1, b_2, \ldots, b_n), \text{max}(d_1, d_2, \ldots, d_n)\rangle$$

The cost of the connector (n_1, n_3, n_4) in Fig. 5*b* will be $\langle 7, 3\rangle$, since the costs of connectors (n_1, n_3) and (n_1, n_4) are respectively $\langle 7, 2\rangle$ and $\langle 10, 3\rangle$:

$$\circ(\langle 7, 2\rangle, \langle 10, 3\rangle) = \langle 7, 3\rangle$$

To simplify Fig. 5*b*, we inserted only the costs for the 1-connectors, but the costs for the other connectors can be easily computed with the $\circ$ operation, and are all reported in Table 3.

So far, we are able to translate an entire network with QoS requirements in a corresponding AND-OR weighted graph, but still we need some algebraic framework to model our preferences for the links to use in the best tree. For this reason, we use the semiring structure (Sec. 4). An exhaustive explanation of the semiring framework approach for shortest-distance problems is presented in [14].

For example, if we are interested in maximizing the bandwidth of the distribution tree, we can use the c-semiring $S_{Bandwidth} = \langle \mathcal{B} \cup \{0, +\infty\}, \text{max}, \text{min}, 0, +\infty\rangle$ (otherwise, we could be interested in minimizing the global bandwidth with $\langle \mathcal{B} \cup \{0, +\infty\}, \text{max}, \text{min}, +\infty, 0\rangle$. We can use $S_{Delay} = \langle \mathcal{D} \cup \{0, +\infty\}, \text{min}, \text{max}, +\infty, 0\rangle$ for the delay, if we need to minimize the maximum delay that can be experienced on a single link. With this result and the depth of the final tree, we can compute an upper bound for the end-to-end delay. Elements of $\mathcal{B}$ and $\mathcal{D}$ can be obtained by collecting information about the network configuration, the current traffic state and technical information about the links. Since the composition of c-semirings is still a c-semiring [3],

$$S_{Network} = \langle \langle \mathcal{B} \cup \{0, +\infty\}, \mathcal{D} \cup \{0, +\infty\}\rangle, +', \times', \langle 0, +\infty\rangle, \langle +\infty, 0\rangle\rangle$$

where $+'$ and $\times'$ correspond to the vectorization of the $+$ and $\times$ operations in the two c-semirings: given $b_1, b_2 \in \mathcal{B} \cup \{0, +\infty\}$ and $d_1, d_2 \in \mathcal{D} \cup \{0, +\infty\}$,

$$\langle b_1, d_1\rangle +' \langle b_2, d_2\rangle = \langle \text{max}(b_1, b_2), \text{min}(d_1, d_2)\rangle$$

$$\langle b_1, d_1\rangle \times' \langle b_2, d_2\rangle = \langle \text{min}(b_1, b_2), \text{max}(d_1, d_2)\rangle$$

Clearly, the problem of finding best distribution tree is multi-criteria, since both bandwidth and delay must be optimized. We consider the criteria as independent among them, otherwise they can be reconducted to a single criteria. Thus, the multidimensional costs of the connectors are not elements of a totally ordered set, and it may be possible to obtain several trees, all of which are not *dominated* by others, but which have different incomparable costs.

For each destination node, the costs of its outgoing 0-connector will be always included in every path reaching the node. As seen in Section 3, a 0-connector ha only one input node but no destination nodes. If we consider a receiver as a plain node, we can set the cost as the $\mathbf{1}$ element of the adopted c-semiring ($\mathbf{1}$ is the unit element for $\times$), since the costs to reach this node are already totally described by the other connectors ending in this node: practically, we give highest QoS values to this 0-connector, infinite bandwidth and null delay. Otherwise we can imagine a receiver as a more complex subnetwork, and thus we can set the cost of the 0-connector as the cost needed to finally reach a node in that subnetwork (in case we do not want, or cannot, show the topology of the subnetwork).

Figure 5 shows the transformation of the network of Fig. 1 into a corresponding AND-OR graph. Group members not interested in the communication are not represented, since the distribution tree has not to reach them. In Fig. 5*a*, one receiver (node n_9) has been replaced with a subnetwork, with respect to Fig. 1.

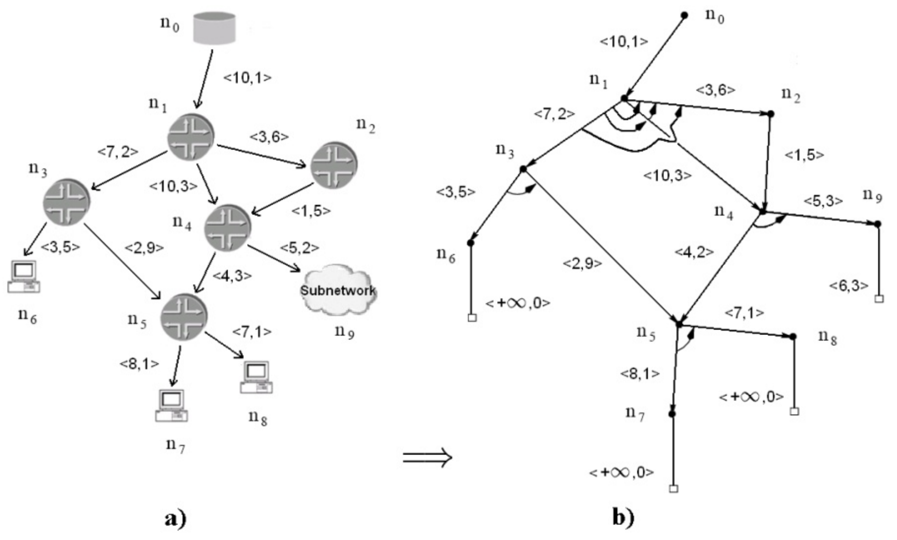

Fig. 5: A network example and the corresponding AND-OR graph representation.

6 AND-OR graphs using SCLP

In this section we will show how to represent an AND-OR graph as an SCLP program over a specific c-semiring. First of all, we present and solve the problem on a AND-OR graph whose connectors have unidimensional costs (monocriteria), and after we propose a multi-criteria example concerning the multicast QoS network in Fig. 5*b*.

To represent the classical problem where the meaning of *best* tree is the tree whose *sum* of the costs of its connectors is *minimum*, we consider an SCLP program over the semiring $S = \langle N, \min, +, +\infty, 0\rangle$, which, as noted above, is an appropriated framework to represent constraint problems where one wants to minimize the sum of the costs of the solutions. In the first example, we are interested in finding the tree with the minimum cost. Thus, the constant associated to each connector, that is, its cost, is in N and the costs of the subtrees rooted at the sons of a certain node are *combined* together using the $+$ operator. More precisely, to describe the operation of combination we will use the $\times$ operator of the semiring; the additive operator will be useful instead to compare different trees, since the partial order $\leq_S$ is induced by $+$ operation.

From the weighted AND-OR graph problem in Fig. 2 we can build an SCLP program as follows. For each connector we have two clauses: one describes the connector and the other one its cost. More precisely, the head of the first clause represents the starting node n_{i_0}, and its body contains both the final nodes and a predicate, say $f_{i_0,\ldots,i_k}$, representing the cost of the connector from node n_{i_0} to nodes $n_{i_1},\ldots,n_{i_k}$. Then, the second clause is a fact associating to predicate $f_{i_0,\ldots,i_k}$ its cost (which is a semiring element).

Table 2: The SCLP program representing all the AND trees over the weighted AND-OR graph problem in Figure 2.

n_0 :- $f_{0,5,4}$, n_5, n_4.	$f_{0,5,4}$:- 2.
n_0 :- $f_{0,1}$, n_1.	$f_{0,1}$:- 3.
n_1 :- $f_{1,2}$, n_2.	$f_{1,2}$:- 3.
n_1 :- $f_{1,3}$, n_3.	$f_{1,3}$:- 7.
n_2 :- $f_{2,3}$, n_3.	$f_{2,3}$:- 1.
n_2 :- $f_{2,5,4}$, n_5, n_4.	$f_{2,5,4}$:- 3.
n_3 :- $f_{3,6,5}$, n_6, n_5.	$f_{3,6,5}$:- 1.
n_4 :- $f_{4,5}$, n_5.	$f_{4,5}$:- 2.
n_4 :- $f_{4,8}$, n_8.	$f_{4,8}$:- 2.
n_5 :- $f_{5,6}$, n_6.	$f_{5,6}$:- 3.
n_5 :- $f_{5,7,8}$, n_7, n_8.	$f_{5,7,8}$:- 2.
n_6 :- $f_{6,7,8}$, n_7, n_8.	$f_{6,7,8}$:- 3.
n_7 :- 4.	n_8 :- 3.

The whole program corresponding to the AND-OR graph problem in Figure 2 can be seen in Table 2. To solve the AND-OR graph problem it is enough to perform a query in the SCLP framework; for example, if we want to compute the cost of the best tree rooted at n_2 and having as leaves a subset of the nodes representing the receivers (in this case, $\{n_7, n_8\}$), we have to perform the query n_2. The operational semantics machinery finds all trees (modulo some cuts due to heuristics) and then combines all the solutions via the additive operation of the semiring, which in this case is min.

Notice that to represent classical best tree problems in SCLP, we do not need any variable. Thus the resulting program is propositional. However, this program, while giving us the cost of the best tree, does not give us any information about the connectors which form such a tree. This information could be obtained by providing each predicate with an argument which represents the connector chosen at each step, as we did for shortest path problems in [6].

The formulation given in Table 2 has also another drawback: the resulting best tree has not the constraint to reach all the leaves representing the receivers. To take in account this problem, we provide a different SCLP formulation (example in Table 3): we provide the clauses to represent sub-trees S_{tree}(x,y), where x is the root and y is the list of the tree leaves. The tails for this clause can be both $C(x, y)$ or $C(x, z), S_{tree}(z, y)$, since a sub-tree can directly be represented by a connector ($C(x, y)$) with input node x and a list y of output nodes, or a connector reaching intermediate nodes z plus a sub-tree from z to destination nodes y (tail $C(x, z), S_{tree}(z, y)$). At last, a clause with head $S_{tree}(x|y, z)$ and tail $S_{tree}(x, z_1), S_{tree}(y, z_2), append(z_1, z_2, z)$ is needed to manage the junction of the disjoint trees with roots in the list $[x|y]$. Clauses with head $C(n_i, [n_k, ...n_k])$ represent the connectors of the AND-OR graph with input node n_i and the list of output nodes $n_k, ...n_k$, while the tail specifies the cost of the connector. To properly use this formulation, we suppose the list of the output nodes of the connectors as lexicographically ordered. With the query $S_{tree}(n_r, [n_0, ..., n_k])$, we are able to find the *best tree* rooted at n_r and reaching all the leaves in the list $[n_0, ..., n_k]$.

An example of this kind of program is given in in Table 3, and corresponds to the AND-OR graph problem in Figure 5*b*. Performing the query $S_{tree}(n_0, [n_6, n_7, n_8, n_9])$, the semantics of this program is represented in Figure 6 and corresponds to the best distribution tree with respect to bandwidth and delay metric values of the links. The algebraic cost model has been proposed in Section 5 and is represented by the c-semiring $S_{Network} = \langle\langle \mathcal{B} \cup \{0, +\infty\}, \mathcal{D} \cup \{0, +\infty\}\rangle, +', \times', \langle 0, +\infty\rangle, \langle +\infty, 0\rangle\rangle$. The cost of the tree in Fig. 6 is $\langle 4, 5\rangle$, since $\times'$ computes the *minimum bandwidth - maximum delay* of the connectors.

The final cost of the tree obtained with the SCLP program is equivalent to the one that can be computed using $\times'$ inside the f_r function given in Sec. 3. Starting from source node n_0 and connector (n_0, n_1) with cost $\langle 10, 1\rangle$, the cost of the tree c_{n_0} is

$$c_{n_0} = f_r(c_{n_1}) = \langle 10, 1\rangle \times' c_{n_1}$$

Table 3: The SCLP program representing the weighted AND-OR graph problem in Figure 5*b*.

$S_{tree}(x,y)$	:- $C(x,y)$.		
$S_{tree}(x,y)$	:- $C(x,z)$, $S_{tree}(z,y)$.		
$S_{tree}(x\|y,z)$	:- $S_{tree}(x,z_1)$,$S_{tree}(y,z_2)$, append(z_1,z_2,z).		
C(n$_0$,[n$_1$])	:- $\langle 10,1\rangle$.	C(n$_1$,[n$_2$,n$_3$,n$_4$])	:- $\langle 3,6\rangle$.
C(n$_1$,[n$_3$,n$_4$])	:- $\langle 7,3\rangle$.	C(n$_1$,[n$_2$,n$_4$])	:- $\langle 3,6\rangle$.
C(n$_1$,[n$_2$,n$_3$])	:- $\langle 3,6\rangle$.	C(n$_1$,[n$_2$])	:- $\langle 3,6\rangle$.
C(n$_1$,[n$_3$])	:- $\langle 7,2\rangle$.	C(n$_1$,[n$_4$])	:- $\langle 10,3\rangle$.
C(n$_2$,[n$_4$])	:- $\langle 1,5\rangle$.	C(n$_3$,[n$_5$,n$_6$])	:- $\langle 2,9\rangle$.
C(n$_3$,[n$_5$])	:- $\langle 2,9\rangle$.	C(n$_3$,[n$_6$])	:- $\langle 3,5\rangle$.
C(n$_4$,[n$_5$,n$_9$])	:- $\langle 4,3\rangle$.	C(n$_4$,[n$_5$])	:- $\langle 4,2\rangle$.
C(n$_4$,[n$_9$])	:- $\langle 5,3\rangle$.	C(n$_5$,[n$_7$,n$_8$])	:- $\langle 7,1\rangle$.
C(n$_5$,[n$_7$])	:- $\langle 8,1\rangle$.	C(n$_5$,[n$_8$])	:- $\langle 7,1\rangle$.
C(n$_6$,[ ])	:- $\langle \infty,0\rangle$.	C(n$_7$,[ ])	:- $\langle \infty,0\rangle$.
C(n$_8$,[ ])	:- $\langle \infty,0\rangle$.	C(n$_9$,[ ])	:- $\langle 6,3\rangle$.

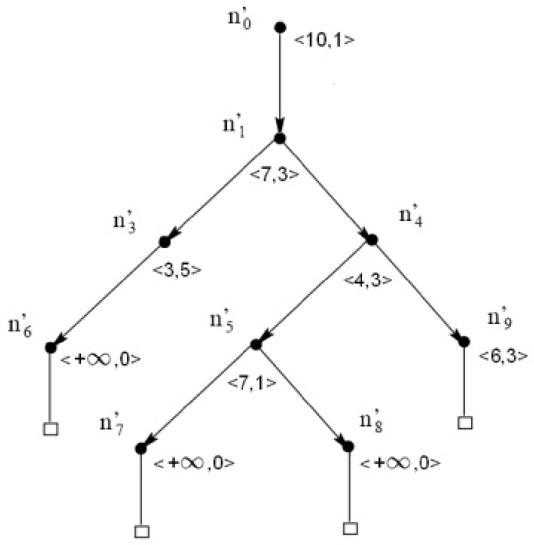

Fig. 6: The best multicast distribution tree corresponding to the program in Table 3.

7 Conclusions

We have described a method to represent and solve the multicast QoS problem with the combination of AND-OR graph and SCLP programming: the best tree on a AND-OR graph correspond to the best multicast distribution tree modelled by the graph. The best tree optimizes some objectives regarding QoS performance, e.g. minimizing the global bandwidth consumption or reducing the delay. The structure of a *c-semiring* defines the algebraic framework to model the costs of the links, and SCLP framework describes and solves the SCSP problem (the best tree) in a declarative fashion. Since several distinct criteria must be all optimized (the costs of the arcs include different QoS metric values), the best tree problem belongs to multi-criteria problem class.

Future research could address also the remodelling of the best tree due to the continuous network-state changes, including the requests of multicast group members to dynamically join in and leave from the communication, or the modifications of the QoS metric values on the links, since a network must be efficiently used to transport multiple flows at the same time.

If the problem is seen as a SCSP, α-*consistency* [1] can be used to speed-up the search in the solution space, by pruning those solutions inconsistent with the receivers requirements.

Even if in this paper we have applied SCLP programs over AND-OR graph to find the best multicast distribution tree, the same framework could be used also to solve problems on decision tables [17], by translating them into AND-OR graphs, or even other *dynamic programming* problems. Decision tables are widely used in many data processing applications for specifying which action must be taken for any condition in some exhaustive set. Every condition is characterized by some combination of the outcomes of a set of condition tests. An important problem is to derive from a given decision table a decision tree which is optimal in some specified sense.

Previous works (such as [12]) introduced a general model of dynamic programming based on AND-OR graphs and showed that each dynamic programming problem could be reduced to the problem of finding an optimal solution tree in an AND-OR graphs. Therefore, SCLP can express the semantic of many of these problems.

References

1. S. Bistarelli. *Semirings for Soft Constraint Solving and Programming*. Volume LNCS 2962. Springer, 2004.
2. S. Bistarelli, U. Montanari and F. Rossi. *Constraint Solving over Semirings*. In *Proceedings of IJCAI'95*, Morgan Kaufman, 1995.
3. S. Bistarelli, U. Montanari and F. Rossi. *Semiring-based Constraint Solving and Optimization*. *Journal of ACM*, vol. 44, no. 2, March 1997.
4. S. Bistarelli, U. Montanari, and F. Rossi. *Semiring-based Constraint Logic Programming*. In *Proc. IJCAI97*. Morgan Kaufman, 1997.
5. S. Bistarelli, U. Montanari and F. Rossi. *SCLP Semantics for (Multi-Criteria) Shortest Path Problems*. Informal Proc. CP-AI-OR'99, Ferrara, Italy, 1999.
6. S. Bistarelli, U. Montanari and F. Rossi. *Soft Constraint Logic Programming and Generalized Shortest Path*. Journal of Heuristics, Kluwer Academic Publishers, 8: 2541, 2002
7. S. Chen, K. Nahrstedt and Y. Shavitt. *A QoS-Aware Multicast Routing Protocol*. IEEE JSAC, vol. 18, no. 12, December 2000.
8. T. H. Cormen, C. E. Leiserson and R. E. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1997.
9. Y. Georget and P. Codognet. *Compiling semiring-based constraints with clp(fd,s)*. In *Proc. CP98, LNCS 1520*. Springer, 1998.
10. J. Jaffar and M.J. Maher. *Constraint Logic Programming: A Survey*. *Journal of Logic Programming*, vol. 19 and 20, 1994.
11. L. Kou, G. Markowsky and L. Berman. *A Fast Algorithm for Steiner Tree*. Acta Informatica, pp. 141145, 1981.
12. A. Martelli and U. Montanari. *From dynamic programming to search algorithms with functional costs*. Proc. Fourth Int. Joint Conf. Artif. Intell. Tbilisi, Sept. 1975, pp. 345-350.
13. A. Martelli and U. Montanari. *Optimizing decision trees through heuristically guided search*. CACM, Dec. 1978, vol.21, n.12.
14. M. Mohri. *Semiring Frameworks and Algorithms For Shortest-Distance Problems*. Journal of Automata, Languages and Combinatorics, 7 (2002) 3, pp. 321-350.
15. J. Moy. *OSPF Version 2*. RFC 2178, July 1997.
16. N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga, 1980.
17. U. W. Pooch. *Traslation of decision tables*. ACM Computing Surveys 6, 2 (1974), 125-151.
18. G. N. Rouskas and I. Baldine. *Multicast Routing with End-to- End delay and Delay Variation Constraints*. IEEE JSAC, pp. 346356, April 1997.
19. B. Wang and J.C. Hou. *Multicast routing and its QoS extension: problems, algorithms, and protocols*. IEEE Network, Vol.14, No.1, Jan./Feb., 2000.
20. P. Winter. *Steiner Problems in Networks: A Survey*. Networks Volume 17, Issue 2, Pages 129-167, Summer 1987.
21. X. Xiao and L.M. Ni. *Internet QoS: A Big Picture*. IEEE Network, 13(2):8-18, Mar 1999.
22. O. Younis and S. Fahmy. *Constraint-Based Routing in the Internet: Basic Principles and Recent Research*. IEEE Communications Surveys and Tutorials, Vol. 5 No. 1, 2003.