# A Constraint Logic Programming based Decision Support System for Network Traffic Management

**G. Martini, C. Moiso, M. Porta**

**CSELT**
**Centro Studi E Laboratori Telecomunicazioni**

**via Reiss Romoli 274, 10148 Torino (Italy)**

**Ph.: +39-11-2286780; Fax: +39-11-2286909**

**Abstract**

One of the problems in network traffic management is to select the correct actions to improve the network performance in order to control network congestions caused by external factors, such as faults or overloads. The controls to apply are decided by the network operators that must hold a lot of data and requirements in consideration.
This paper presents an experimental decision support system, whose purpose is to assist the network operators in the selection of the set of expansive controls called Temporary Alternative Routing (TAR) that is the best one w.r.t. a given optimum criterion. The system has been realized adopting an Advanced Information Processing technique that allows to integrate symbolic rules and linear program models: the Constraint Logic Programming (CLP) paradigm.
The implementation language is a new CLP language, called VEL (i.e. Constraints And Logic in Italian), whose peculiarity is a set of optimization primitives well integrated with the CLP paradigm.
At present, the system is under testing as a component of the network management module in a simulated environment.

**Keywords**: Network Traffic Management, Constraint Logic Programming.

## 1. Introduction

A telephone network is generally planned in order to support normal busy-period loads. An external factor, such as a fault or an overload, can cause a serious congestion state, that must be recovered through the application of management actions in order to maintain an acceptable grade of service of the network. The aim is reached through the application of expansive actions, that increase the network resources, or through protective actions, that decrease the offered traffic, for example, limiting at the origin node the call attempts of a relation.

At present, the network management is executed by expert operators at the traffic management centre. Specialized procedures have the task to signal the possible abnormal behaviours, while the operators decide the more convenient set of actions to be applied w.r.t. the management principles adopted.

The task of the operators is very hard, because they must take into account a lot of static and dynamic aspects (such as the network topology, the current traffic loads, the current call routing, the available resources,...) and various requirements depending on the management strategy adopted by the telephone company and the CCITT recommendations. Moreover, the changes in the topology and trunk capacities, performed during the periodical updates of the network, can invalid the previous experience.

The operators must act so that the selected actions do not cause congestions in other part of the network, or they do not block an excessive traffic quantity. In order to assist them, several attempts to develop Decision Support Systems (DSS) based on Advanced Information Processing (AIP) techniques have been made in the last years.

We have implemented such a DSS adopting an approach of an AIP technique, the Constraint Logic Programming (CLP), that integrates the rule-based approaches and the linear model-based ones, in order to merge their advantages and overcome their drawbacks.

This paper describes both the adopted technique and the main aspects of the developed system. Section 2 introduces the environment for which our DSS was designed. Section 3 describes the characteristics of CLP, and briefly introduces the main features of VEL [Moiso92], the CLP language we designed. In section 4, we describe the DSS implemented in VEL, and in section 5, we report some of the results of the testing of the system performed in a simulated environment.

## 2. The application scenario

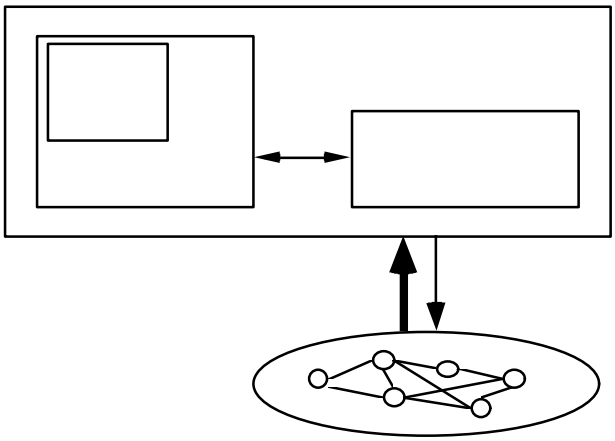
The telephone network we considered is a hierarchical one, with two levels of nodes (terminal nodes and transit exchanges), with a hierarchical, step-by-step routing control. We assumed that the exchanges in the network are able to perform the following commands, through which it is possible to recover serious congestion states:

- selective tar command:
  tar n <i,$n_1$,...,$n_k$,j>
  its execution adds the path <i,$n_1$,...,$n_k$,j> (where $n_1$,...,$n_k$ are the k transit node of the expansive route) as the n-th route in the routing table of the relation (i,j); the tar command can have an additional parameter, the percentage of call attempts offered that can overflow on the expansive route;
- selective skip command
  skip i,j n
  that cancels the n-th route in the routing table of the relation (i,j).

The network behaviour is monitored by a supervisor system that provides a set of network parameters, updated at fixed time periods, and performs alarm detection

functionalities. Some of the data provided by this monitor module (and used by the DSS we developed) are: trunk congestion alarms, the traffic offered to a relation, the trunk loss probabilities, an estimation of the resources available at each trunk in the following interval.

The data provided by the monitor module are used by the network operators to detect the network congestion states and to decide the more convenient set of actions to apply w.r.t. the adopted traffic management strategy in order to recover them. The strategy can be (partially) automatized by a traffic control module, placed in the network supervisor system between the monitor module (which collects data and detects alarms) and the traffic operators. The DSS we developed is a component of such a traffic control module based on the traffic management strategy described in [Buttò89].



The DSS we realized concerns the selection of the set of relations to be expanded and the actions to be applied in order to optimize the network performance. The decision is made according to some principles and an optimization criterion.

The principles try to keep the controls simple and reduce the effects on the other relations. Some of the principles introduced in the developed DSS are:

- an expansive control for a relation consists of the introduction in its routing table of a TAR, that diverges as soon as possible from the planned routes;
- the expansive route has only one node (called the pivot node of the TAR) not present in the planned routing table.

The criterion used to choose the set of actions among all the possible solution is introduced as a cost function to be optimized (i.e. maximized or minimized): different function single out different objectives of the telephone company, such as the carried traffic or the number of rejected calls.

## 3. An Advance Information Processing approach for traffic control

In the last years, several attempts have been made to apply AIP techniques in the development of systems that assist the network supervisors in selecting the correct controls to handle network difficulties and improve network performance.

The different technologies are characterized by the kind of the adopted knowledge acquisition and representation:

1) rule-based formalism: several expert systems [Jimenez88,Georgeff90] were developed through the synthesis of the knowledge provided by the experts into if-then symbolic rules, which are processed by an inference engine;
2) neural network approach: the controls to be performed are learned through training algorithm on a sensible set of examples [Silver90];

3) mathematical models: the network behaviour and the action effects are described through a system of equations/disequations, solved through operational research techniques (e.g. the simplex method) [Warfield88].

All these approaches have their advantages but also some drawbacks. One of the disadvantages in 1) is the difficulty to synthesize knowledge from human experts, by using only if-then rules; moreover it is hard to include optimization criteria in pure rule-based reasoning. In 2) it is difficult to produce from examples on one network rules valid for a class of different network topologies.

The last approach seems to be quite promising because the behaviour of a generic network is formalized through a mathematical model, that can be easily instantiated to a particular network. But such a model is not able to represent other aspects of the problem, such as constraints on the kind and/or the number of the actions to be performed or principles ruling the control activities. In fact, it is quite difficult to code as numerical constraints all the conditions and the rules involved into the definition of a strategy to control network congestions.

The approach we have adopted is based on an integration of rules and mathematical models: in this way we mix the so-called deep knowledge of the system with the so-called shallow knowledge.

The advantage of the integration of the two knowledge representation techniques is that each piece of knowledge in the system can be coded in the most suitable way: knowledge about the behaviour of the network and quantitative conditions can be easily described through a mathematical model, while the heuristic rules, that determine the control strategy, synthesized from the domain experts can be expressed through symbolic logic formulae. Moreover changes and extensions of the knowledge in the system can be performed very quickly, because there is an almost one-to-one (declarative) correspondence between the knowledge and its coding.

On the other hand, there are some limits in the class of mathematical models we can use to model the network behaviour, that impose us to find a compromise between the accuracy of the model and its computational characteristics.

In fact, it is well-known that the network behaviour has non linear models (see, for instance, the B-Erlang formula), but, unfortunately, there are no generic tools to solve non-linear systems and, then, to perform optimizations w.r.t. them. Generic tools are instead available for linear systems (of equations/disequations), both to solve them and to optimize linear functions w.r.t. them: an example of such tools is the simplex method.

### 3.1 Constraint logic programming and the language VEL

In the last years, a new computation paradigm, the constraint logic programming (CLP), has been developed [VanHentenryck89], as an extension of logic programming, suitable for the representation and the computation of both numerical and symbolic knowledge. Several extensions of Prolog, among which PrologIII [Colmerauer87], CLP(R) [Jaffar87], CHIP [Dincbas88], has been designed according to the new paradigm: the search-based symbolic computation mechanism of Prolog (based on the resolution inference rule) is enriched with the possibility of dealing with numerical conditions (in the general case, conditions in an algebra). When in the computation of a branch of the search tree a constraints must be resolved, it is added to the set of those previously met in the same branch:

- a special solver algorithm is invoked to check the consistency of the new set of (numerical) constraints (and possibly to compute its solution);
- if the solver detects an inconsistency the current branch fails, and the next one is tried (according to the Prolog operational semantics).

The CLP paradigm is parametric w.r.t the domain of constraints. It can be instantiated, according to the application requirements, with the suitable kinds of constraints (e.g. constraints on real number, on integer number, or on boolean values): each constraint domain requires the development of a constraint solver algorithm to be invoked during the computation.

The CLP paradigm is not just a simple mix of numerical and symbolic computations; in fact it has several additional advantages:

- the two forms of conditions can be present in the same rule: there is a real integration of the two form of knowledge and not just a coupling of them;
- the numerical constraints can be added in an incremental way: this is in contrast with the operational research tools that require that the set of constraints is provided as a whole;
- disjunctive numerical constraints can be handled: the operational research tools are not able to cope with this kind of constraints in an efficient way.

All these properties have to be efficiently supported by the adopted solver algorithms: they must be incremental and integrated with the mechanism implementing the Prolog search-based computations.

We have designed and implemented a new member of the CLP language family, called VEL (*Vincoli E Logica*, i.e. Constraints and Logic in Italian), an extension of Prolog with the facility to cope with constraints on rational numbers and on domains of natural numbers, that provides a rich set of primitives to performs optimizations in an efficient way. VEL is similar to the well-known language PrologIII [Colmerauer87] w.r.t the constraints on rational numbers and the primitives *one_value*, *max_value* and *min_value*, and to CHIP [Dincbas88] w.r.t the primitives to deal with constraints on domain variables.

The peculiarity of VEL consists in the incremental optimization primitives *maximize* and *minimize*: they have the semantics similar to *max_value* and *min_value* (which compute the maximum/minimum value of an objective function w.r.t. the current set of constraints on rationals), with the difference that the objective function specified by *maximize*, or *minimize*, is internally recorded, and during the continuation of the computation, each time a new constraint is inserted, the solver continues to compute the maximum or minimum value of the objective function, besides to verify the consistency of the constraint set. The only, but considerable, advantage to use *maximize* (resp. *minimize*) instead of *max_value* (resp. *min_value*) is a more efficient computation, because *maximize* (resp. *minimize*) computes the new optimal value of the objective function starting from the current one, while *max_value* (resp. *min_value*) starts from scratch the computation. The current optimal value is unified with the argument of the primitive *curr_value*. These primitives are based on an incremental version of the simplex algorithm, derived from the dual simplex theory.

VEL provides two other primitives:

*max(G,T,Max)*
*min(G,T,Min)*

where *G* is a goal, *T*, *Max* and *Min* are terms, whose variables must occur in G. The semantics of *max* (resp. *min*) is to compute all the solutions of the goal *G* in order to find the solution that maximizes (resp. minimizes) the (numerical) value of *Max* (resp. *Min*).

The search space produced by *max* (resp. *min*) can be shrunk, without loosing the best solution, through the pruning of those branches that cannot generate a solution better than the current one because the approximation of the value that *Max* (resp. *Min*) would have w.r.t them is lower (resp. greater) than the current optimal value, computed by a previously explored branch. In order to perform this pruning, VEL supplies with the

primitive *comp_curr_val(X)*, that succeeds when either there is not yet a first solution or *X* is better than the current optimal value.

The primitives *max* and *min* are independent of numeric constraints, but can be used in a very efficient way with *maximize* and *minimize* to search optimal solutions in a context of disjunctive constraints on rational numbers. The pruning process could be based on the monotone relationship among a linear constraint set *Eq* and the optimal value *m* of a linear objective function *f* w.r.t. *Eq*: if *m'* is the optimal value of *f* w.r.t *Eq È C* (where *C* is a new set of constraints), then *m'* is not better than *m*. This property implies that the (partial) optimal value of an objective function *f* w.r.t a partial set of constraints of a problem *P* is an approximation greater (resp. lower) than the final optimal value of *f* w.r.t. the complete set of the constraints of *P*, and can be efficiently used in the comparison with the current final optimal value of *f* through the primitive *comp_curr_val* in order to shrunk the search space.

A program scheme could be:

*r(...) :- ...,maximize(Z,E), max((p(...,R),curr_value(Max)),R,Max),...*

while, according to the programmer, it is advantageous put the following check in the body of the clauses defining *p* after the insertion of a set of constraints $c_1,...,c_n$:

*q(...) :- ...,{*$c_1$*,...,*$c_n$*},curr_value(Partial_Max),comp_curr_val(Partial_Max),...*

One of the application fields we had in mind when we designed VEL was the development of near real-time systems. Therefore, we paid attention to an efficient implementation of VEL. VEL programs are compiled into an extension of the abstract machine for Prolog, where the elaboration of the numerical constraints is tightly integrated with the implementation of the Prolog functionalities. The solver and the optimization algorithms are implemented in C.

A detailed description of VEL and its inference engine (based on an extension of the consistency check algorithm in [Jaffar87], the simplex algorithm and its incremental version) is reported in [Moiso91, Moiso92].

## 4. The decision support system

We used VEL to develop an experimental DSS that suggests to the network supervisors the expansive controls to improve the network performance during serious congestion states. The system is based on the approach, that integrates the rule-based and the model-based ones, described in the previous section. In the following are given some details about the logic of the system.

The developed DSS receives as input a list LEXP of relations that could need expansive controls. The list LEXP (which is computed according to the strategy in [Buttò89]) contains the relations (i,j) that could suffer from congestion inside the toll network, but whose call attempts successfully routed to the destination node j have a high probability of reaching the user. These relations are characterised through the following conditions:

- unacceptable end-to-end blocking probability;
- good ASR (answer seizure rate) in the destination node.

The output produced by the DSS is a sequence of tar commands on a subset of LEXP. The computation of the action sequences requires some further data on the network current status (e.g. routing tables, parameters, alarms,...) that are got from the monitor system.

The selection of the relations in LEXP to be expanded and the choice of the actions to be activated are based on different kinds of coded knowledge, and can be affected by the operators through a user(-friendly) interface.

The processing performed by the support system can be sketched in the following steps:

1) input of the list LEXP;
2) filtering of the list LEXP, by performing some correlation among relations in it and the congestion state of their final trunks and transit exchanges;
3) assignment to each relation in the filtered LEXP of a control scheme, according to a set of rules that implement some principles of the control strategy (developed by the domain experts); each scheme is parametric w.r.t. a set of possible TARs;
4) selection of the relations to be actually expanded and choice for each of them of the TAR to be performed, according to a linear model and a (cost) function to be optimized;
5) output of the sequence of actions according to the control schemes of the selected relations, instantiated with the chosen TARs.

In the rest of this section we briefly analyse the main aspects of each steps.

Due to the variability of measurements, because of the stochastic nature of the traffic, it is necessary to perform some correlations among different parameters in order to make safer the detection of abnormal behaviours. As the considered network is hierarchical, the final trunks are good indices of the network behaviour; therefore LEXP is filtered to remove from it all the relations whose final trunks and exchanges are not congested, in order to focus the controls only on those relations that actually suffer.

In the third step, each relation (i,j) in the filtered LEXP is associated to a control scheme and a set of possible pivots for the TAR (called pivot(i,j) in the following). A control scheme is a template that describes the structure of the controls to be applied to a relation. It contains the following data, that specify the arguments for the tar command:

- the position in the routing table where to insert the TAR;
- the template of the TAR;
- the traffic offered to the TAR.

The template of the TAR is parametric w.r.t. the pivot node: it is the sequence of the nodes of the alternative route, where the position of the pivot node is shown by the character *. The template of the TAR of the relation (i,j) can be instantiated with one of the nodes in pivot(i,j) to provide a possible TAR for (i,j): the actual pivot of the TAR is selected in the optimization phase in step 4).

The control schemes are assigned to the relations according to a set of rules. The assigned scheme depends on the following parameters:

- the current routing table of the relation;
- the congestion state of the final trunks and transit exchanges;

In some cases the assignment depends also on the available pivot nodes.

The pivot(i,j) set is built according to the topology of the network and to the state of the trunks; if

<i,t,*,j>

is the template of the TAR in the control scheme for the relation (i,j), then a transit node n is put in the pivot(i,j) set if the following conditions hold:

- n is a transit exchange that does not occur in the current routing table of (i,j);
- there exist the trunks t->n and n->j (called the pivot trunks);
- n, t->n and n->j are not congested;
- the trunks t->n and n->j have residual capacities available.

The residual capacities of a trunk t is an estimation of the circuits of t that will be free during the following time period.

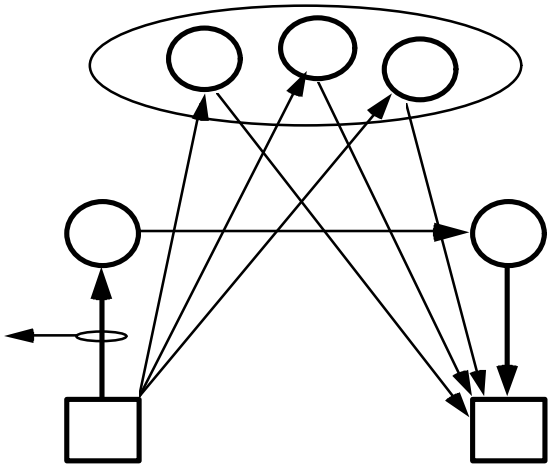
If pivot(i,j) is empty, the relation (i,j) is removed from the list of relations to be expanded.

Let us consider the portion of a hierarchical network in the figure, where I and J are two terminal nodes (connected only through the final route) and Ci and Cj are their respective transit exchanges. An example of rule to assign a control scheme is:

if
1) in the routing table of the relation (I,J) there is only the final route
and
2) the trunk I->Ci is in congestion
then
the control scheme for (I,J) is:
- position of the TAR: 1;
- template of the TAR: <I,*,J>
- traffic offered to the TAR: traffic offered to (I,J) * loss prob. of I->J

The set {h,k,y} is the set of nodes that satisfy the previous conditions for pivot(I,J).

These rules are coded by exploiting the logic features of VEL.



In the step 4), the system realizes the decisional part by selecting the relations to be actually expanded and, for each of these, the pivot node of the TAR in the relative pivot set.

These choices are driven by a model of the traffic flows and a function (to be maximized) that introduces a criterion of best solution, i.e. the solution that optimizes the network performance (according to the aspects covered by the adopted cost function definition).

The model is a linear one, where the trunks are traffic limiters, similar to that introduced in [Warfield88], but with disjunctive linear constraints. It introduces the following variables, denoting traffic flows, for each relation (i,j) to be expanded:

- $A'_{ij}$:
  the total traffic carried by the expansion of (i,j);
- $X_k(i,j)$, where $k \in$ pivot(i,j):
  the traffic carried by the possible TAR for (i,j) with k as pivot.

For each relation (i,j), the quantity $A'_{ij}$ must verify the following constraints:

- $A'_{ij} \geq 0$ and $A'_{ij} \leq A_{ij}$:

i.e. the traffic carried by the TAR of (i,j) can not exceed $A_{ij}$, that is the traffic offered to the expansion of (i,j);

- $A'_{ij}=\Sigma_{k \in pivot(i,j)} X_k(i,j)$.

For each pivot trunk we introduce a constraint, whose meaning is: the sum of the (expanded) traffic routed through the trunk can not exceed the amount of the residual resources of the trunk. This set of constraints limits the amount of the (expanded) traffic that a trunk can carry without damaging the other relations.

The function to be maximized is the total expanded traffic: $\Sigma_{(i,j)} A'_{ij}$.

At the moment, we assume that for each relation there is at most one TAR. The program tries (in an intelligent and efficient way) all the possible pivot nodes for each relation (including the possibility of selecting none of them). This is modelled through a set of disjunctive constraints.

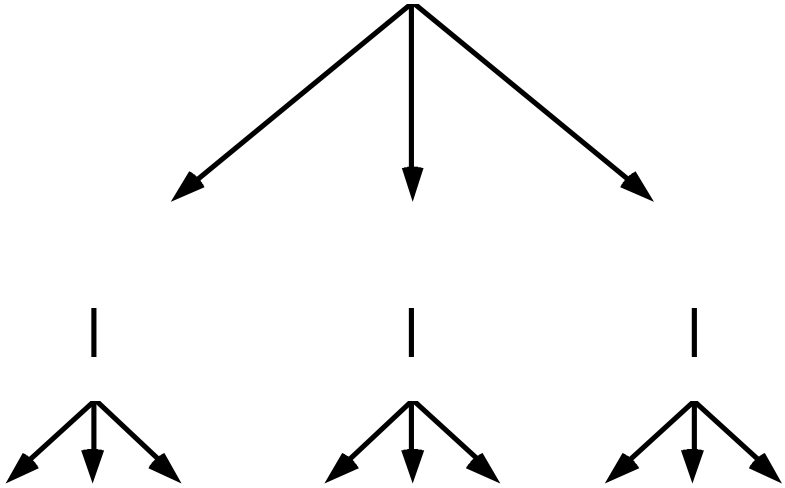
The selection of k as pivot for the relation (i,j) is modelled by the following set of constraints:

- $X_h(i,j)= 0$, where $h \in$ pivot(i,j) and h_k:
  only the route through k can carry the traffic of (i,j) to be expanded;
- $X_k(i,j) \geq p*A_{ij}$:
  the TAR through k must carry at least a percentage p of the offered traffic.

The decision of not expanding a relation (i,j) is modelled by:

- $X_h(i,j)= 0$, for each $h \in$ pivot(i,j)

The constraints are disjunctive because different choices of the pivot (explored in different branches of the search space) are modelled through different sets of constraints.



The numerical constraints are integrated with additional conditions, such as a limit on the number of TARs to be activated.

At the end, the program produces the solution that maximizes the cost function. In addition for each selected TAR the model computes the amount of traffic $A'_{ij}$ that it can carry without damaging the other relations. There may be some relations for which no pivot is chosen: in fact, we have to perform a global optimization, whose result (i.e. the best solution) is not just the sum of the best TAR of each relation.This step is implemented by exploiting the features of VEL to perform in efficient way optimizations in a search space, constrained by both numerical and symbolic conditions.

The control schemes associated to the relations are combined with the result of this step to produce the set of actions to be activated. The actions are performed only on

those relations for which a pivot is chosen; the selected pivot and the amount of the carried traffic are used to build the parameters of the tar commands:

- the pivot node instantiates the template of the TAR to form the path of the route;
- the percentage of the TAR is computed from the amount of the carried traffic.

The DSS has a user interface to interact with the network operator. The operators can require different solutions computed according to different sets of parameters (e.g. the maximum number of TARs), compare them and, then, select the one that they prefer.

## 5. Experimental results in a simulated environment

The DSS that has been developed keeping in mind the necessity of an easy porting into a real system and use as a component of the management system of a hierarchical telephone network, has been tested within a simulated environment [Malabocchia91], on a hierarchical network of 16 terminal nodes and 4 transit exchanges.

The tests have been conducted under different scenarios, i.e. with different failures of the trunks, overloads and random-generator seeds. The effectiveness of the controls proposed by the DSS has been verified by comparing the histories of the simulations with the controls and those not having any applied control at all.

Moreover, in order to minimize the effects of the stochastic variability of the telephone traffic, we have focused only over the loss probability of each relation; more precisely, we have classified the relations into 2 classes, depending on having been interested by an anomaly (e.g., if there is a failure on the trunk 18/17 all the relations having in the own nominal routing plan the trunk 18/17 belong to this class), or not.

For each one of these classes, we have considered the mean values of the loss probability, computed for all the relations belonging to the class, over the whole time of a simulation. The following general results may be drawn:

- the simulations without any controls show high values for the loss probability of the interested relations, whereas the others have normal values: this means that the effects of the anomalies are local;
- the application of expansive controls (as suggested by the DSS) has bettered the performance of the interested relations, without affecting the others: this reveals that the controls don't spread out the congestion all over the network;
- the positive trend of the loss probability function (after the application of expansive controls) implies a similar trend for the total expanded traffic, which is in perfect accordance with the objective function.

The previous results confirm the validity of the strategy used by the DSS, moreover the following issues can be argued on the approach and the strategy:

- by considering the tests on the current network and some hand-made examples corresponding to bigger networks, the system has an execution time suitable to be used as a near real-time decision tool;
- the LEXP list construction and the correlation phase have a crucial role in order to detect the set of relations that actually suffer: a careful tuning of the involved thresholds is required;
- further study is needed to cope with the relations that can not be expanded either at all or in a satisfactory way;
- the CLP paradigm allowed both a rapid development of the system and a quick implementation of extensions and changes of the strategy, the mathematical model and the heuristics used to shrink the search space (without loosing the optimal solution).

## 6. References


[Buttò89] **M. Buttò, G. Giacobbo Scavo**, Network management policies: which aims and how to pursue them, Proc. Network Management and Control Workshop (1989).

[Colmerauer87] **A. Colmerauer**, Une introduction à PrologIII, Annales de Télécommunication 44, n. 5-6 (1989) 229-241.

[Dincbas88] **M. Dincbas, et al.**, The constraint logic programming language CHIP, in Proc. FGCS'88 (1988), 693-702.

[Georgeff90] **M. Georgeff, A. Rao**, Intelligent real-time network management, Proc. 10th Int. Workshop on Expert Systems and their Applications (1990), 87-101.

[Jaffar87] **J. Jaffar, S. Michaylov**, Methodology and implementation of a CLP system, in Proc. 4th Conf. on Logic Programming (MIT Press, 1987), 196-218.

[Jimenez88] **S. Jimenez**, The application of ES to Network Traffic Management in Telecom Australia, Telecomm. Journal of Australia, vol. 38, n. 1 (1988), 25-33.

[Malabocchia91] **F. Malabocchia, et al.**, A real-time simulation tool for network management applications, Proc. SimTec'91 (1991).

[Moiso91] **C. Moiso, M. Porta**, VEL: a constraint logic language for the integration of symbolic and numerical knowledge, Proc. 6th Italian Conf. on Logic Programming (1991), 365-375.

[Moiso92] **C. Moiso, M. Porta**, Optimizations in Constraint Logic Programming, submitted to publication (1992).

[Silver90] **B. Silver**, Netman: a learning network traffic controller, GTE Labs Report.

[VanHentenryck89] **L. VanHentenryck**, Constraint satisfaction in logic programming (MIT Press, 1989).

[Warfield88] **R. Warfield, D. McMillan**, A linear program model for automation of network management, IEEE J. on Selected Areas in Comm. 6, n. 4 (1988), 742-750.