Disjunction of Non-Binary and Numeric Constraint Satisfaction Problems

Miguel A. Salido, Federico Barber

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia
Camino de Vera s/n, 46071
Valencia, Spain
{msalido, fbarber}@dsic.upv.es

Abstract. Nowadays, many researchers are working on Constraint Satisfaction Problems (CSPs). Many CSPs can be modelled as non-binary CSPs and, theoretically, they can be transformed into an equivalent binary CSP, using some of the current techniques. However, this transformation may be an inadequate or inefficient way to manage certain types of non-binary constraints. In this paper, we propose an algorithm called *DHSA* that solves numeric non-binary CSPs with disjunctions in a natural way, as non-binary disjunctive CSP solver. This proposal extends the class of *Horn constraint*, originally studied by Koubarakis, since DHSA manages disjunctions of linear inequalities and disequations with any number of inequalities per disjunction. This proposal works on a polyhedron whose vertices are also polyhedra that represent the non-disjunctive problems. This non-binary disjunctive CSP solver translates, in a preprocess step, the disjunctive problem into a non-disjunctive one by means of a statistical preprocess step. Furthermore, a Constraint Ordering Algorithm (COA) classifies the resultant constraints from the most restricted to the least restricted one. This preprocess step can be applied to other disjunctive CSP solvers in order to find a solution earlier.

Keywords: CSPs, non-binary constraints, Disjunctive constraints.

1 Introduction

Over the last few years, many researchers have been working on Constraint Satisfaction Problems (CSPs), as many real problems can be efficiently modelled as constraint satisfaction problems (CSPs) and solved using constraint programming techniques. Some examples are scheduling, planning, machine vision, temporal reasoning, medical expert systems and natural language understanding. Most of these problems can be naturally modelled using non-binary (or n-ary) constraints, that involve any number of variables. In the constraint satisfaction literature, the need to address issues regarding non-binary constraints has only recently started to be widely recognized. Research has traditionally focused on

binary constraints (i.e., constraints between pairs of variables) [17]. The basic reasons are the simplicity of dealing with binary constraints compared to non-binary ones and the fact that any non-binary constraint satisfaction problem can be transformed into an equivalent binary one [9]. However, this transformation has several drawbacks:

- Transforming a non-binary into a binary CSP produces a significant increase in the problems' size, so the transformation may not be practical [3] [6]. The translation process generates new variables, which may have very large domains, causing extra memory requirements for algorithms. In some cases, solving the binary formulation can be very inefficient [1].
- A forced binarization generates unnatural formulations, which cause extra difficulties for constraint solver interfaces with human users [4].

When trying to solve a problem with non-binary constraints, we are faced with a crucial modelling decision. Do we convert the problem into a binary one or do we leave it in its non-binary representation? If we convert the problem into a binary one, then we can use some of the widely studied algorithms and heuristics for binary constraints to solve the problem. If we leave the problem in its non-binary representation, we have to use algorithms and heuristics for non-binary constraints. Such algorithms and heuristics have not been studied extensively. Our objective is to study the disjunctive non-binary CSP in its natural representation.

Disjunctions of linear constraints over real values are important in many applications [5]. The problem of deciding consistency for an arbitrary set of disjunctions of linear constraints is NP-complete [16].

In [7], Lassez and McAloon studied the class of *generalized linear constraints*, this includes linear inequalities (e.g., $x_1+2x_3-x_4 \leq 4$) and disjunctions of linear disequations (e.g., $3x_1 - 4x_2 - 2x_3 \neq 4 \vee x_1 + 3x_2 - x_4 \neq 6$). They proved that the problem consistency for this class can be solved in polynomial time.

Koubarakis in [5] extends the class of *generalizad linear constraints* to include disjunctions with an unlimited number of disequations and *at most one* inequality per disjunction. (e.g., $3x_1-4x_2-2x_3 \leq 4 \vee x_1+3x_2-x_4 \neq 6 \vee x_1+x_3+x_4 \neq 9$). This class is called *Horn constraints*. He proved that deciding consistency for this class can be done in polynomial time.

In this paper, we propose an algorithm for solving problems with numeric non-binary disjunctive constraints. This algorithm called "Disjunctive Hyper-polyhedron Search Algorithm" (DHSA) manages non-binary disjunctive CSPs in a natural way as a non-binary CSP solver. This proposal extends the class of *Horn constraint* originally studied by Koubarakis [5] since DHSA manages disjunctions of linear and non-linear disequation and linear inequalities with any number of inequalities per disjunction. The objective of our non-binary CSP is more ambitious since besides of deciding consistency, DHSA obtains the minimal domains of the variables and the solutions that the user requires. This algorithm carries out the search through a polyhedron that maintains the problem inequalities in its faces and updates itself when new constraint is studied. This proposal

overcomes some of the weaknesses of other proposals like *disjunctive Forward-checking* and *disjunctive Real Full Look-ahead* keeping its complexity unchanged when the domain size and the number of disequations increase.

2 Preliminaries

Briefly, a numeric constraint satisfaction problem $P = (X, D, C)$ is defined by:

- a set of variables $X = \{x_1, x_2, ..., x_n\}$;
- a set of domains $D = \{D_1, ..., D_n\}$ where each variable $x_i \in X$ has a set D_i of possible values (its domain);
- a set of constraints $C = \{c_1, c_2, ..., c_p\}$ restricting the values that the variables can simultaneously take.

A solution to a CSP is an assignment of a value from its domain to every variable such that all constraints are satisfied. The objective in a CSP may be determining:

- whether a solution exists, that is, if the CSP is consistent;
- all solutions or only one solution, with no preference as to which one;
- the minimal variable domain;
- an optimal, or a good solution by means of an objective function defined in terms of certain variables.

2.1 Notation and definitions

Definition 1[8]
Let *P=(X,D,C)* be a numeric CSP, P is globally consistent if and only if $\forall x_i \in X$, $\forall a \in D_i$, $x_i = a$ belongs to a solution to P.

We will summarize the notation that will be used in this paper.

Generic: The number of variables in a CSP will be denoted by n. The constraints will be denoted by c with an index, for example, c_1, c_i, c_k, Also, all constraints are global constraints, that is, all constraints have the maximum arity n.

Variables: To represent variables, we will use x with an index, for example, x_1, x_i, x_n.

Domains: The domain of the variable x_i will be denoted by $D_i = [l_i, u_i]$, so that the domain length of the variable x_i is $d_i = u_i - l_i$. We assume continuous domains for variables.

Constraints: Let $X = x_1, ..., x_n$ be a set of real-valued variables. Let α be a polynomial of degree 1 (i.e. $\alpha = \sum_{i=1}^{n} p_i x_i$) over X, and let b, p_i be real numbers. A *linear relation* over X is an expression of the form $\alpha r b$ where $r \in \{<, \leq, =, \neq, \geq, >\}$. Specifically, a *linear disequation* over X is an expression of the form $\alpha \neq b$ and a *linear equality* over X is an expression of the form $\alpha = b$.

In accordance with previous definitions, the constraints that we are going to manage are linear relations of the form:

$$Inequalities : \sum_{i=1}^{n} p_i x_i \leq b \tag{1}$$

$$Disequations : \sum_{i=1}^{n} p_i x_i \neq b \tag{2}$$

$$Non-linear \;\; Disequations : F(x) \neq b \tag{3}$$

where x_i are variables ranging over continuous intervals and $F(x)$ is a non-linear function. Using the above constraints, equalities can be written as conjunctions of two inequalities. Similarly, strict inequalities can be written as the conjunction of an inequality and a disequation. Thus, we can manage all possible relations in $\{<, \leq, =, \neq, \geq, >\}$.

2.2 Constraints

Traditionally constraints are considered *additive*, that is, the order of imposition of constraints does not matter, all that matter is that the conjunction of constraints be satisfied [2]. Our framework will manage internally the constraints in an appropriate order with the objective of reducing the temporal and spatial complexity.

The *arity* of a constraint is the number of variables that the constraint involves. A *unary* constraint is a constraint involving one variable. A *binary* constraint is a constraint involving a pair of variables. A non-binary constraint is a constraint involving an arbitrary number of variables. When referring to a non-binary CSP, we mean a CSP where some or all of the constraints have an arity of more than 2. DHSA is a CSP solver that manages non-binary constraints.

Example. The following are examples of atomic non-binary constraints that DHSA can manage:

$2x_1 - 5x_2 + 3x_3 - 9x_4 \leq 4$, $3x_1^4 + 6\sqrt[3]{x_5} - 2x_4^3 \neq 9$, $x_1 - 2x_2 - 4x_3 + x_4 < 4$

The first and second constraints are managed directly by DHSA. The last constraint is transformed into two constraints:

$x_1 - 2x_2 - 4x_3 + x_4 < 4 \Rightarrow x_1 - 2x_2 - 4x_3 + x_4 \leq 4 \wedge x_1 - 2x_2 - 4x_3 + x_4 \neq 4$

In this paper, we assume a non-binary CSP where variables are bounded in continuous domains (for example: $x_i \in [l_i, u_i]$) and a collection of non-binary constraints of the form (1)(2) and (3).

3 Specification of DHSA

DHSA is considered to be a CSP solver that manages non-binary disjunctive constraints. In Figure 1, a general scheme of DHSA is presented. Initially, DHSA studies the significant parameters such as number of variables and number of disjunctive inequalities. Depending on these parameters DHSA runs a preprocess step in which two algorithms are carried out: a heuristic process to translate the disjunctive problem into a non-disjunctive one, and the *Constraint Ordering Algorithm* (COA) to classify the most restricted constraints first, reducing the temporal complexity considerably. Then, using the resultant ordered and non-disjunctive problem, DHSA carries out the consistency study with the resultant problem as a classic CSP solver.

3.1 Preprocess Step

Solving disjunctive constraint problems requires considering an exponential number of non-disjunctive problems. For example, if the problem has k disjunctive constraints composed by l atomic constraints, the number of non-disjunctive problems is l^k.

DHSA uses a preprocessing heuristic technique to obtain the non-disjunctive problem that is likely to satisfy the problem. This technique can be compared with the sampling from a finite population, in which there is a population, and a sample is chosen to represent this population. In this context, the population is the convex hull of all solutions generated by means of the Cartesian Product of variable domain bounds. This convex hull may be represented by a polyhedron with n dimensions and 2^n vertices. However, the sample that the heuristic technique chooses is composed by n^2 vertices of the complete polyhedron[1]. These vertices are well distributed in order to represent the entire population.

With the selected sample of vertices (n^2), the heuristic technique studies how many vertices $v_{ij} : v_{ij} \leq n^2$ satisfy each atomic constraint c_{ij} . Thus, each atomic constraint c_{ij} is labelled with p_{ij}: $c_{ij}(p_{ij})$, where $p_{ij} = v_{ij}/n^2$ represents the probability that c_{ij} satisfies the whole problem. Thus, the heuristic technique selects, the atomic constraint with the highest p_{ij} for each disjunctive constraint.

As we remarked in the preliminaries, constraints are considered *additive*, that is, the order in which the constraints are studied does not make any difference [2]. However, DHSA carries out an internal ordering of the constraints. If some constraints are more restricted than others, these constraints are studied first in order to reduce the resultant polyhedron. Thus, the remaining constraints are more likely to be redundant. However, if the remaining ones are not redundant, they generate less new vertices, so the temporal complexity is significantly reduced.

The constraint ordering algorithm (COA) classifies the atomic constraints in ascending order of the labels p_{ij}. Therefore, DHSA translates the disjunctive

[1] The heuristic selects n^2 items if $n > 3$, and 2^n vertices, otherwise

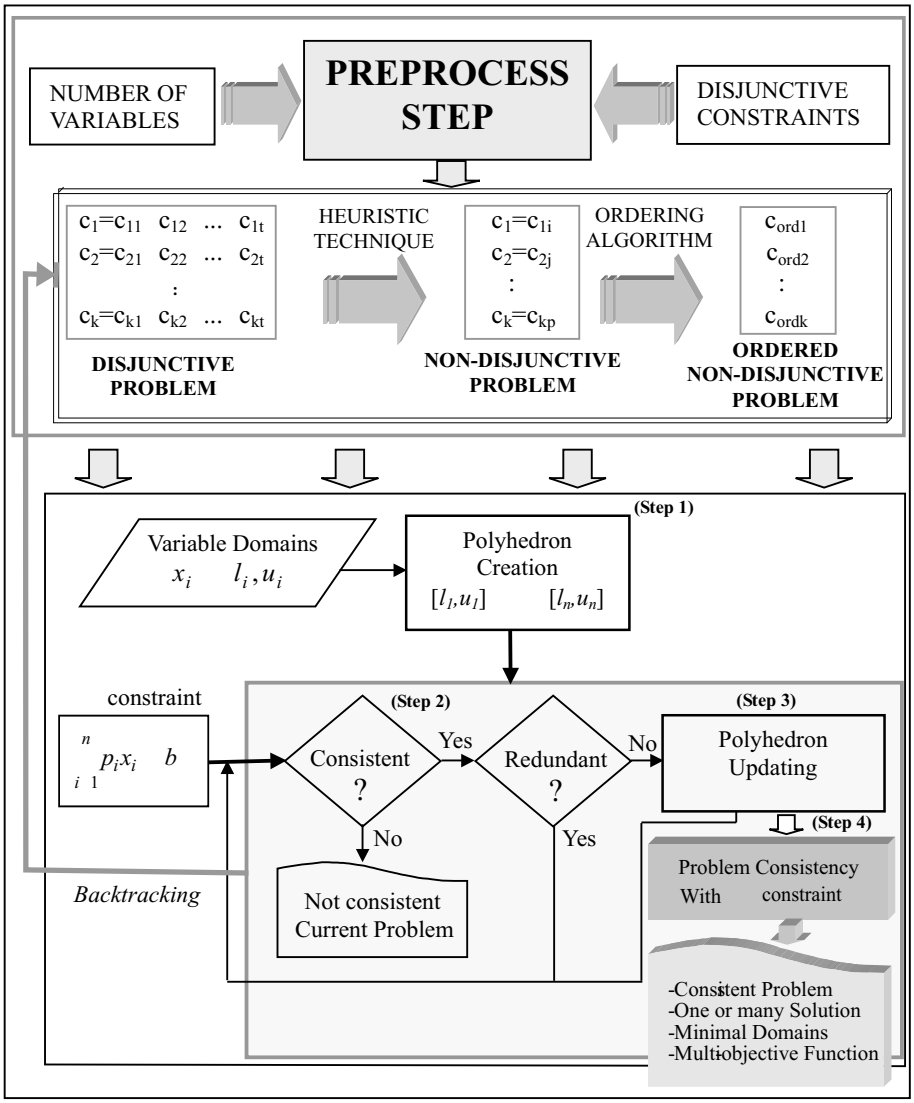

Fig. 1. General Scheme of DHSA

non-binary CSP into a non-disjunctive and ordered CSP in order to be studied by the CSP solver.

In Figure 2, we can observe an example in which the atomic selected constraints in a disjunctive CSP are: (c_{12}, c_{21}, c_{32}). Therefore, DHSA will run the corresponding non-disjunctive problem. Let suppose that the atomic constraint labels are $(c_{12}(3), c_{21}(2), c_{32}(1))$. If DHSA carries out the consistency study in the order of imposition of constraints (option 1) (c_{12}, c_{21}, c_{32}), DHSA will generate

Fig. 2. Example of the Constraint Ordering Algorithm

6 new vertices. However, if DHSA runs the ordering algorithm, which classifies the constraints in ascending order (option 2) (c_{32}, c_{21}, c_{12}), DHSA will generate only one new vertex with the corresponding time reduction.

If the selected and ordered non-disjunctive problem is not consistent, the algorithm backtracks and the the heuristic technique selects the following set of atomic constraints that is more likely to satisfy the problem.

3.2 CSP solver

The CSP solver used by DHSA is a complete CSP solver [11] [13]. It generates an initial polyhedron (step 1) with 2^n vertices created by means of the Cartesian product of the variable domain bounds $(D_1 \times D_2 \times ... \times D_n)$. DHSA classifies the selected constraints (by the preprocess step) in two different sets: the inequality set and the disequation set. For each $(\leq)$ constraint, the CSP solver carries out the consistency check (step 2). If the $(\leq)$ constraint is not consistent, the CSP solver returns *not consistent current problem* and it backtracks to the preprocess step in order to select a new non-disjunctive problem. If the constraint is consistent, the CSP solver determines whether the $(\leq)$ constraint is not redundant, and updates the polyhedron (step 3), i.e. the CSP solver eliminates the

inconsistent vertices and creates new ones. Finally, when all inequalities have been studied, DHSA studies the consistency with the disequations. Therefore, the solutions to CSP are all vertices, and all convex combinations between any two vertices that satisfy all disequations.

DHSA can obtain some important results such as: the problem consistency; one or many problem solutions; the minimal domain of the variables; the vertex of the polyhedron that minimises or maximises some objective or multi-objective function.

Theorem 1. The CSP solver is sound and complete.

Proof: The CSP solver is sound and complete because it always maintains the solution set in a convex polyhedron whose faces are the problem ($\leq$) constraints. If the resultant polyhedron is not empty, each solution found by the CSP solver is correct, and all solutions can be found into the convex hull of the resultant polyhedron.

Proposition 1. By theorem 1, DHSA obtains global consistency in each non-disjunctive problem.

4 Analysis of the DHSA

DHSA spatial cost is determined by the number of vertices generated. Initially, in the preprocess step, DHSA studies the consistency of the n^2 vertices with the atomics constraints, where n is the number of problem variables. Thus, the spatial cost is $O(n^2)$. Then, DHSA generates 2^n vertices. In the worst case, for each ($\leq$) constraint (step 2), DHSA might eliminate only one vertex and generate $n + c_{\leq}$ new vertices, where $c_{\leq}$ is the number of previously studied ($\leq$) constraints. Thus, the number of vertices is $2^n + k(n + c_{\leq})$, where k is the number of disjunctive constraints. Therefore, the spatial cost is $O(2^n)$.

The temporal cost can be divided into five steps: Preprocess, initialization, consistency check with ($\leq$) constraints, actualization and consistency check with ($\neq$) constraints. The preprocess cost is $O(ktn^2)$ where t is the maximum number of atomic constraints in a disjunctive constraint. The initialization cost (step 1) is $O(2^n)$, because the algorithm generates 2^n vertices. For each ($\leq$) constraint (step 2), the consistency check cost depends linearly on the number of polyhedron vertices, but not on the variable domains. Thus, the temporal cost is $O(2^n)$. Finally, the actualization cost (step 3) and the consistency check with ($\neq$) constraints depend, on the number of vertices, that is $O(2^n)$. In the worst case, if all non-disjunctive problems must be checked, these three steps must be carried out l^k. Thus, the temporal cost in the worst case is: $O(ktn^2) + O(2^n) + l^k(k \cdot (O(2^n) + O(2^n)) + O(2^n)) \implies O(l^k 2^n)$. Note, that in practice this complexity is much smaller because the heuristic technique obtains statistically the more appropriate non-disjunctive problem at the preprocess step, so it is not necessary to try all possibilities.

5 Evaluation of the Polyhedron Search Algorithm

In this section, we compare the performance of DHSA with some of the more current CSP solvers. We have selected Forward-checking [4] (FC) and Real Full Look-ahead [10] (RFLA)[2] because they are the most appropriate techniques that can manage this CSP typology. We have used a PIII-800 with 256 Mb. of memory and Windows NT operating system.

Generally, the benchmark sets are used to test algorithms for particular problems, but in recent years, there has been a growing interest in the study of the relation between the parameters that define an instance of CSP in general (i.e., the number of variables, domain size, density of constraints, etc..).

In this empirical evaluation, each set of random constraint satisfaction problems was defined by the 5-tuple $< n, c_{\leq}, c_{\neq}, d, t >$, where n was the number of variables, $c_{\leq}$ the number of disjunctive ($\leq$) constraints, $c_{\neq}$ the number of ($\neq$) constraints, d the length of variable domains and 't' the number of atomic constraints for each disjunctive ($\leq$) constraint. The problems were randomly generated by modifying these parameters. We considered all constraints as global constraints, that is, all constraints had maximum arity. Thus, each of the graphs shown sets four of the parameters and varies the other one in order to evaluate the algorithm performance when this parameter increases. We tested 100 test cases for each type of problem and each value of the variable parameter, and we present the mean CPU time for each of the techniques. Four graphs are shown which correspond to the four significant parameters (Figures 3, 4, 5, 6). The domain length parameter is not significant for DHSA, so we do not include this graph. Each graph summarizes the Mean CPU time for each technique. Here, for unsolved problems in 200 seconds, we assigned a 200-second run-time. Therefore, these graphs contain a horizontal asymptote in $time = 200$.

In Figure 3, the number of variables was increased from 3 to 11, the number of ($\leq$) and ($\neq$) constraints, the variable domain length and the number of atomic constraints were set $< n, 6, 20, 2000, 6 >$ respectively. The graph shows a global view of the behaviour of the algorithms. The mean CPU time in FC and RFLA increased faster than DHSA. When the unsolved problems were set to time=200 and the others maintained their real-time cost, we observed that FC was worse than RFLA. However, DHSA always had a better behaviour and was able to solve all the problems satisfactorily.

In Figure 4 , the number of variables, the number of ($\neq$) constraints, the variable domain length and the number of atomic constraints were set $< 11, c, 40, 2000, 6 >$, and the number of random ($\leq$) constraints ranged from 2 to 10. The graph shows that the mean CPU times in FC and RFLA increased exponentially and were near the horizontal asymptote for problems with 10 ($\leq$) constraint. The number of unsolved problems increased in FC and RFLA much more than in DHSA.

[2] Forward-checking and Real Full Look-ahead were obtained from CON'FLEX, which is a C++ solver that can handle constraint problems with continuous variables with disjunctive constraints. It can be found in: http://www-bia.inra.fr/T/conflex/Logiciels/adressesConflex.html.

Fig. 3. Mean CPU Time when the number of variables increased

Fig. 4. Mean CPU Time when the number of inequalities increased

In Figure 5, the number of variables, the number of ($\leq$) constraints, the variable domain length and the number of atomic constraints were set $< 11, 6, c, 2000, 6 >$, and the number of random ($\neq$) constraints ranged from 10 to 1000. The graph shows that the behavior of FC and RFLA got worse when the number of ($\neq$) constraints increased. DHSA did not increase its temporal complexity due to the fact that it carried out the consistency check of the ($\neq$) constraints in low complexity. The number of unsolved problems was very high for both FC and RFLA, while DHSA had a good behavior. Note that DHSA was proved with an amount of 10^5 disequations and it solved them in few seconds (< 3 sc.)

In Figure 6, the number of variables, the number of ($\leq$) and ($\neq$) constraints and the variable domain length were set $< 10, 6, 10, 200, t >$, and the atomic constraints were increased from 4 to 14. To study the behaviour of the algorithms when the number of atomic constraints increased, we chose $t - 1$ non-consistent atomic constraints and only one consistent atomic constraint. That is, if the number of atomic constraints was 8, the random constraint generator generated 7 non-consistent atomic constraints and 1 consistent constraint.

Fig. 5. Mean CPU Time when the number of disequations increased

Fig. 6. Mean CPU Time when the number of atomic constraints increased

Thus, we could observe the behaviour of the algorithm when the number of atomic constraints increased. FC and RFLA had worse behaviour than DHSA. DHSA makes a preprocess step in which it selects the most appropriate non-disjunctive problem. Also, this preprocess step is made in polynomial time, so the temporal cost is very low.

We present a comparison between DHSA without the Constraints Ordering Algorithm (N-COA) and DHSA with the Constraints Ordering Algorithm (Y-COA) in Table 1. This comparison was carried out in two different contexts: in non-consistent problems and in consistent problems. It can be observed that the number of vertices generated was higher in DHSA N-COA than DHSA Y-COA in both cases due to the fact that the Constraints Ordering Algorithm first selects the more appropriate non-disjunctive problems.

Following, we present a comparison between RFLA and FC with the proposed preprocess step (Y-RFLA and Y-FC) and without it (N-RFLA and N-FC) in Table 2. The random generated constraints had the following properties: the number of variables, the number of ($\neq$) constraints, the variable domain length

	Non-Consist Prob.		Consist Prob.	
Const	N-COA	Y-COA	N-COA	Y-COA
2	560	0	1050	588
5	1500	420	2920	870
10	2800	872	4215	1625
15	3200	1054	6538	2135
20	3520	1314	7346	2627

Table 1. Number of vertices generated in problems $< 9, c, 40, 100, 10 >$

and the number of atomic constraints were set $< 5, c, 5, 10, 2 >$ and the number of $(\leq)$ constraints were increased from 5 to 30. We can observed that the preprocess step reduced the temporal cost in both algorithms. This temporal cost would be reduce if the number of atomic constraints was higher than 2.

	Number of disjunctive constraints				
Algor.	5	10	15	20	30
N-FC	0.3	9.33	22.7	45.4	75.6
Y-FC	0.15	4.5	18.7	32.1	59.7
N-RFLA	0.25	11.3	25.5	41.2	62.5
Y-RFLA	0.2	8.7	14.5	20.1	43.2

Table 2. Mean CPU time in solved problems $< 5, c, 5, 10, 2 >$

6 Conclusion

In this paper, we have proposed an algorithm called DHSA that solves non-binary disjunctive CSP solver. This proposal extends the class of *Horn constraint* originally studied by Koubarakis [5] since DHSA manages disjunctions of linear and non-linear disequation and linear inequalities with any number of inequalities per disjunction. The objective of our non-binary CSP solver may be: to obtain the problem consistency; to get the minimal domains of the variables and to obtain the solutions that the user requires.

This proposal carries out a consistency study using an algorithm composed of two preprocess algorithms. The first algorithm translates the disjunctive problem into a non-disjunctive one and the other algorithm orders the atomic constraints in an appropriate form. Then, a complete algorithm is carried out over the resulting problem in order to study the consistency of the non-disjunctive problem. DHSA overcomes some weaknesses of other algorithms because its behavior is independent from the domain size and the number of atomic constraints, while other approaches depend exponentially on the number of variables, the number of constraints and the domain size.

Currently, we are working on a framework that is dynamically configured depending on the parameters. This framework is composed by DHSA as a complete CSP solver helped by heuristics such as OFHH (a linear heuristic) [14] and POLYSA (a cubic heuristic) [12]. Furthermore, we are applying these techniques to discrete CSPs [15].

7 Acknowledgments

This paper has been partially supported by grant UPV-20010980 from the Technical University of Valencia and grant DPI2001-2094-C03-03 from the Spanish government.

References

1. F. Bacchus and P. van Beek, 'On the conversion between non-binary and binary constraint satisfaction problems', *In proceeding of AAAI-98*, 311–318, (1998).
2. R. Bartk, 'Constraint programming: In pursuit of the holy grail', *in Proceedings of WDS99 (invited lecture), Prague, June*, (1999).
3. C. Bessire, 'Non-binary constraints', *In Proc. Principles and Practice of Constraint Programming (CP-99)*, 24–27, (1999).
4. C. Bessire, P. Meseguer, E.C. Freuder, and J. Larrosa, 'On forward checking for non-binary constraint satisfaction', *In Proc. Principles and Practice of Constraint Programming (CP-99)*, 88–102, (1999).
5. M. Koubarakis, 'Tractable disjunction of linear constraints', *In Proc. 2nd International Conference on Principles and Practice of Constraint Programming (CP-96)*, 297–307, (1999).
6. J. Larrosa, *Algorithms and Heuristics for total and partial Constraint Satisfaction*, Phd Dissertation, UPC, Barcelona, 1998.
7. J.L. Lassez and K. McAloon, 'A canonical form for generalizad linear constraints', *In Advanced Seminar on Foundations of Innovative Software Development*, 19–27, (1989).
8. O. Lhomme, 'Consistency techniques for numeric CSPs', *In International Joint Conference on Artificial Intelligence (IJCAI-93)*, 232–238, (1993).
9. F. Rossi, C. Petrie, and V. Dhar, 'On the equivalence of constraint satisfaction problems', *In proceeding of European Conference of Artificial Intelligence*, 550–556, (1990).
10. D. Sabin and E.C. Freuder, 'Understanding and improving the MAC algorithm', *In proceeding of Principles and Practice of Constraint Programming*, 167–181, (1997).
11. M.A. Salido and F. Barber, 'An incremental and non-binary CSP solver: The Hyperpolyhedron Search Algorithm', *In Proc. of 7th International Conference on Principles and Practice of Constraint Programming (CP-01), LNCS 2239*, 799–780, (2001).
12. M.A. Salido and F. Barber, 'POLYSA: A polinomial algorithm for non-binary constraint satisfaction problems with <= and <>', *In Proceeding of EPIA-2001 Worshop on Constraint Satisfaction and Operation Research (CSOR01)*, 99–113, (2001).

13. M.A. Salido, A. Giret, and F. Barber, 'Constraint satisfaction by means of dynamic polyhedra', *In Operational Research Proceedings 2001, Springer Verlag*, **1**, 405–412, (2001).
14. M.A. Salido, A. Giret, and F. Barber, 'A non-binary constraint satisfaction solver: The One-Face Hyperpolyhedron Heuristic', *Research and Development in Intelligent Systems XVIII, Springer Verlag*, **1**, 313–324, (2001).
15. M.A. Salido, A. Giret, and F. Barber, 'Integration of Discrete and Non-binary CSPs with Linear Programming Techniques', *To appear in Proc. of CP-2002 Workshop on Cooperative Solvers in Constraint Programming*, (2002).
16. E. Sontag, 'Real addition and the polynomial time hierarchy', *Information Processing Letter*, **20**, 115–120, (1985).
17. E. Tsang, *Foundation of Constraint Satisfaction*, Academic Press, London and San Diego, 1993.