On the Dual Representation of Non-binary Semiring-based CSPs *†

Javier Larrosa‡ , **Rina Dechter**
Information and Computer Science Dept.
University of California at Irvine
444 Computer Science Building
Irvine, CA 92697-3425
{javier, dechter}@ics.uci.edu

July 10, 2000

Abstract

It is well known that any non-binary CSP can be reformulated as a binary CSP. In this paper we show that the same translation methods can be applied in the soft constraints framework. We observe that any non-binary soft constraint CSP can be reformulated as a problem with only binary and unary constraints. Interestingly, the translation leads to binary constraints that are hard (define conditions of mandatory satisfaction) and unary constraints that are soft (define a preference criterion among the set of solutions). We elaborate our observation in the semiring-based framework.

Keywords: non-binary constraints, duality, soft constraints, semiring-based CSP.

1 Introduction

Constraint satisfaction problems (CSPs) involve the assignment of a set of variables subject to a set of constraints. They provide an expressive framework allowing the formalization of many interesting problems arising in a variety of domains.

Constraints in the classical CSP model are refered to as *hard* because they have to be necessarily satisfied. In past years, substantial work has been devoted to the extention of the classical CSP framework into a more general one in which it is possible to represent uncertain or partial knowledge. Some of the developed frameworks include *fuzzy* CSPs [11, 10], *probabilistic* CSPs [9], *weighted* CSPs, *hierarchical* CSPs [5]. Constraints in these frameworks have been referred to as *soft* because they express weaker satisfiability requirements.

*This work was funded by the NSF under grant IIS-9610015.

†The first author was also funded by the Spanish CICYT under project TAP1999-1086-C03-03.

‡On leave from *Universitat Politecnica de Catalunya*, Spain.

Soft constraints typically express a preference criterion among different solutions. Recently, it has been shown that most constraint frameworks (including hard and soft constraints) can be expressed under a unifying framework, the so-called *semiring-based CSPs* (SCSPs) [12, 13].

A useful observation in the hard constraints model is that any non-binary CSP (i.e. with constraints involving more than two variables) can be translated into an equivalent binary problem (i.e. all its constraints involve at most two variables). There are two well-known translation methods: the *dual* representation [7] and the *hidden variable* representation [8].

In this paper we make the observation that the duality results on classical CSPs are preserved in soft constraint CSPs. Specifically, we show that it is possible to obtain the dual and the hidden variable representation of a non-binary semiring-based CSP in a similar way to how it is done in a classical CSP. The binary reformulations include both binary and unary constraints, where all binary constraints are hard and all unary constraints are soft. Unary soft constraints associate weights to domain values (i.e. $w_i(x_i)$). In problems having unary soft constraints there is an implicit objective function that combines the different weights (for instance, $f(X) = \sum_{i=1}^{n} w_i(x_i)$).

This translation shows that unary soft constraints provide (at least in theory) a sufficient problem specification language for the framework. Surprisingly, little work has been done in the constraint community on algorithms for this model, while other models have received a lot of attention under the claim that they were, in a way, minimal (for instance Max-CSP [1]). Observe as well that problems with unary optimization criteria have been the focus of a great attention in the closely related fields of *Operations Research* and *Combinatorial Optimization*. We believe that our observation may motivate further work in that direction.

The strucuture of this paper is as follows: In the next Section we present an example of our mapping. This example may be sufficient to convey the principle idea to the reader that is familiar with duality results and soft constraint frameworks. The rest of the paper presents a formal proof in the SCSP framework. Section 3 provides the necessary background of the semiring framework. Section 4 formally presents our mapping. Finally, in Section 5 we give the conclusions of the paper.

2 Examples

2.1 general constraint optimization

Consider a *general constraint optimization problem* defined by a set of variables $X = (x_1, \ldots, x_n)$ taking values over a domain D, a set of constraints $C = (c_1, \ldots, c_p)$ and a set of cost functions $F = (f_1, \ldots, f_q)$. Each constraint has a scope (denoted $var(c_i)$) and specifies the allowed combinations of values for variables in $var(c_i)$. Each cost function also has a scope ($var(f_i)$) and associates a weight with each tuple over $var(f_i)$. The task is:

$$\min_{\{X | X \text{ satisfies } C\}} \{\sum_{i=1}^{q} (f_i(X))\}$$

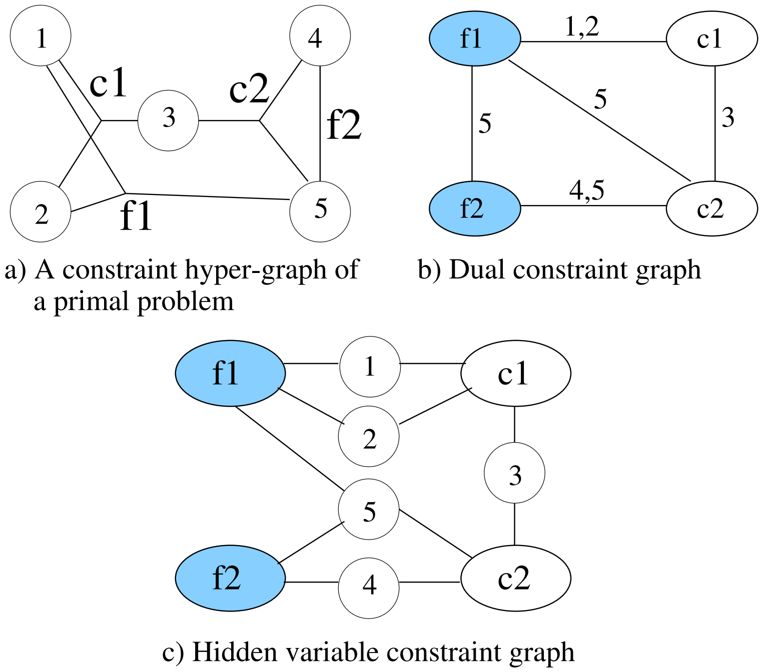

Figure 1: A primal constraint hyper-graph (a) and its corresponding dual (b) and hidden variable graphs (c). Shadowed nodes indicate variables with unary cost functions.

The structure of a constraint problem is depicted by a constraint hyper-graph, whose nodes represent problem variables and hyper-edges describe the scopes of constraints and cost functions. Figure 1.a shows the constraint hyper-graph of an instance of this problem having five variables ($\{1, 2, 3, 4, 5\}$), two constraints (c_1 and c_2, with scopes $\{1, 2, 3\}$ and $\{3, 4, 5\}$, respectively) and two cost functions (f_1 and f_2, with scopes $\{1, 2, 5\}$ and $\{4, 5\}$, respectively).

In the *dual representation* there is one variable for each original constraint $X_C = (x_{c_1}, \ldots, x_{c_p})$ and one variable for each original cost function $X_F = (x_{f_1}, \ldots, x_{f_q})$. Variables associated to constraints take values over the set of tuples that satisfy them ($D_{c_i} = \{t| \text{ tuple over } var(c_i) \text{ that satisfies } c_i\}$). Variables associated to cost functions take values over the cost function domain (i.e. $D_{f_i} = \{t| \text{ tuple over } var(f_i)\}$). There is a binary constraint between every pair of variables that share a variable in their scope (for instance there is a constraint between x_{c_2} and x_{f_1} because $5 \in var(c_2)$ and $5 \in var(f_1)$). These constraints only allow the simultaneous assignment of tuples to variables if the common components of the tuple *match*. There is a unary cost function associated with each variable x_{f_i} that assigns a weight to each element t of D_{f_i}, which is exactly the cost $f_i(t)$. Thus, with that dual formulation, the problem can be expressed as,

$$\min_{\{X_C, X_F | \text{binary constr. are satisfied}\}} \{\sum_{i=1}^{q} f_i(x_{f_i})\}$$

Figure 1.b shows the dual constraint graph of the problem depicted in 1.a

when using hyper-graph representation. Hyper-edges in the hyper-graph become nodes in the dual graph. Edges in the dual graph connect nodes with common elements in their scope. We label the edges with the common elements.

In the *hidden variable representation* there is one variable for each original constraint $X_C = (x_{c_1}, \dots, x_{c_p})$ and one variable for each original cost function $X_F = (x_{f_1}, \dots, x_{f_q})$ as in the dual representation. In addition, there is a variable for each original variable $X = (x_1, \dots, x_n)$. Variables in X_C and X_F are called *hidden*. There is a binary constraint between each pair of hidden-non hidden variable such that the hidden variable has the non hidden variable in its scope (for instance, there is a constraint between x_{c_1} and 3 because $3 \in var(c_1)$). These constraints only allow the simultaneous assignment of values that *match* in their common components. As in the dual representation, there is a unary cost function associated to each variable x_{f_i} that assigns a weight to a domain value t which is $f_i(t)$. Under the hidden variable representation, the problem is defined as,

$$\min_{\{X, X_C, X_F | \text{binary constr. are satisfied}\}} \{\sum_{i=1}^{q} f_i(x_{f_i})\}$$

Figure 1.c shows the hidden variable constraint graph of our example. Nodes and hyper-edges in the hyper-graph (Figure 1.a) become nodes in the hidden variable constraint graph. There is an edge between hidden and non hidden nodes if the corresponding non hidden variable belongs to the scope of the corresponding hidden variable.

2.2 weighted crossword generation

Consider the *weighted crossword generation problem* defined by an alphabet, a dictionary of words of fixed size d and a weight associated to each word in the dictionary. The goal is to fill up a $d \times d$ grid with alphabet letters such that each grid slot (i.e. maximal group of consecutive cells either vertically or horizontally) form a word from the dictionary. Weights indicate that some words are more suitable than others, and the preferred solution is the one that maximizes the sum of word weights.

In the *non-binary* formulation of this problem there is a variables for each grid cell $X = \{x_{ij} | 1 \leq i, j \leq d\}$ and each variable has the alphabet as domain. There is one constraint associated to each grid slot ($C_H = (c_{h_1}, \dots, c_{h_d})$, $C_V = (c_{v_1}, \dots, c_{v_d})$ for horizontal and vertical grids, respectively). Constraints are functions that associate weights to tuples of size d. If the tuple forms a word from the dictionary, it associates the word weight, else it associates a minimal weight (for instance, $-\infty$). The goal is,

$$\max_X \{\sum_{i=1}^{d} c_{h_i}(x_{i1} \dots x_{id}) + \sum_{i=1}^{d} c_{v_i}(x_{1i} \dots x_{di})\}$$

In the *dual* representation there is one variable for each slot ($X_H = (x_{h_1}, \dots, x_{h_d})$ for horizontal and $X_V = (x_{v_1}, \dots, x_{v_d})$ for vertical). Each variable takes values on the dictionary words. There is a binary constraint for each pair of horizontal-vertical slots that only allows pairs of words if they have the same letter in their common cell. In addition, there is a unary cost function for each variable ($C_H = (c_{h_1}, \ldots, c_{h_d})$, $C_V = (c_{v_1}, \ldots, c_{v_d})$)that assign to each domain value the corresponding word weight. The problem is defined as,

$$\max_{\{X_H, X_V | \text{binary constr. are satisfied}\}} \{\sum_{i=1}^{d} c_{h_i}(x_{h_i}) + \sum_{i=1}^{d} c_{v_i}(x_{v_i})\}$$

In the *hidden variable* representation there is one variable for each slot and cell ($X_H = (x_{h_1}, \ldots, x_{h_d})$, $X_V = (x_{v_1}, \ldots, x_{v_d})$ and $X = \{x_{ij} | 1 \leq i, j \leq d\}$). Slot variables take values on the dictionary words and cell variables take values on the alphabet. There is a binary constraint between a pair of slot-cell variables if the cell belongs to the slot. This constraint only allows the assignment of a word to the slot and a letter to the cell if the letter matches with the corresponding letter of the word. There is a unary cost function for each slot variable that assigns to each word the corresponding weight. The problem is defined as,

$$\max_{\{X, X_H, X_V | \text{binary constr. are satisfied}\}} \{\sum_{i=1}^{d} c_{h_i}(x_{h_i}) + \sum_{i=1}^{d} c_{v_i}(x_{v_i})\}$$

3 Preliminaries

In this section we provide the necessary definitions and properties of the SCSP framework. Most of them are extracted from [13]. We start with the definition of *c-semiring*, required to express and combine preferences, level of confidence or any other type of soft constraint.

Definition: (*c-semiring*)

A *c-semiring* is a tuple $(A, +, \times, \mathbf{0}, \mathbf{1})$ such that:

- A is a set and $\mathbf{0}, \mathbf{1} \in A$.
- $+$, called the *additive* operation, is closed, commutative, associative, idempotent, $\mathbf{0}$ is its unit element and $\mathbf{1}$ is its absorbing element.
- $\times$, called the *multiplicative* operation, is closed, commutative, associative, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element.
- $\times$ distributes over $+$.

Property:

The previous axioms allow the definition of a partial order, $\leq$, over the set A. Such partial order is defined as: $a \leq b$ iff $a + b = b$. Intuituvely, $a \leq b$ means that b is *better* than a. Some important properties of the partial ordering are:

- The additive operation is monotonically *increasing* on the partial ordering.
- The multiplicative operation is monotonically *decreasing* on the partial ordering.
- $\mathbf{0} \leq a \leq \mathbf{1}$ for all $a \in A$.

Now we can define a semiring-based constraint satisfaction problem,

Definition: (SCSP)

A *semiring-based constraint satisfaction problem* (SCSP) is a tuple (S, V, D, C) where:

- S is a c-semiring $(A, +, \times, \mathbf{0}, \mathbf{1})$.
- V is an ordered set of variables $(x_1, \ldots, x_n)$
- D is a set of domains, $(D_1, \ldots, D_n)$, where D_j is a finite set of possible values for variable x_j.
- C is a set of constraints $(c_1, \ldots, c_m)$. A constraint c_i over the ordered set of variables $var(c_i) = (x_{i_1}, \ldots, x_{i_{r(i)}})$ is a function $c_i : \prod_{j=1}^{r(i)} D_{i_j} \rightarrow A$.

The following two definitions are needed to establish what is the solution of a SCSP.

Definition: (*constraint combination*, $c_i \otimes c_j$)

Consider two constraints c_i and c_j. Then, their *combination* $c_i \otimes c_j$ is a new constraint c such that $var(c) = (var(c_i) \cup var(c_j))$ and $c(t) = (c_i(t \downarrow_{var(c_i)}) \times c_j(t \downarrow_{var(c_j)}))$ (for all t tuple over $var(c)$).

Definition: (*constraint projection*, $c_i \Downarrow_I$)

Consider a constraint c_i and a set I ($I \subset var(c_i)$). Then, the *projection* of c_i over I, $c_i \Downarrow_I$, is a constraint c such that $var(c) = I$ and $c(t) = \sum_{\{t' \text{ on } var(c_i) - I\}} c_i(t \bowtie t')$ (for all t tuple over $var(c)$).

Definition: (*problem solution*, $Sol(P)$, c_P)

Let P be a SCSP. Its *solution* is a constraint defined as $Sol(P) = \bigotimes_{i=1}^{m} c_i$. Normally we will refer to this constraint as c_P.

The solution of a problem is, therefore, a n-ary constraint that combines the set of original constraints.

Definition: (*tuple consistency level*)

Let P be a SCSP, c_P its solution and t a tuple over $var(c_P)$. The *consistency level* of t is its valuation over the solution constraint, $c_P(t)$. If $c_P(t) > \mathbf{0}$ we say that t is consistent.

Intuitively, the consistency level of a tuple indicates how good the tuple is in the problem. The partial order over A allows for the comparison of different tuples.

Observation:

It is possible to compute the consistency level of a tuple by means of the original constraints, instead of using the problem solution. That is,

$$c_P(t) = c_1(t \downarrow_{var(c_1)}) \times \ldots \times c_m(t \downarrow_{var(c_m)})$$

In most cases, we do not need all the information captured in the problem solution. It is enough to know the best consistency that can be obtained in a problem,

Definition: (*best level of consistency*, $blevel(P)$)

Given a SCSP P, its *best level of consistency* is defined as $blevel(P) = (c_P \Downarrow_{\emptyset})$.

Definition: (*hard* and *soft constraints*)

A constraint c is *hard* iff $c(t) \in \{\mathbf{0},\, \mathbf{1}\}$ for all t over $var(c)$. A constraint c is *soft* iff $\exists t$ over $var(c)$ such that $\mathbf{1} > c(t) > \mathbf{0}$.

4 Translating Non-binary Semiring-based CSPs into Binary Semiring-based CSPs

In classical CSPs, the dual and the hidden representation are equivalent reformulations of a problem, where equivalent means that the set of problem solutions is preserved. In SCSPs, we need to preserve the set of consistent tuples and also maintain their level of consistency. This is done by means of the following definition.

Definition: (*consistency equivalence*)

Two Semiring-based CSPs $P_1 = (S, V_1, D_1, C_1)$ and $P_2 = (S, V_2, D_2, C_2)$ are *consistency equivalent* iff there is a bijection between their sets of consistent solutions such that the consistency valuation of mapped tuples is the same. A direct consequence of this definition is that if P_1 and P_2 are consistency equivalent, then $blevel(P_1) = blevel(P_2)$.

In the following two subsections we show how to obtain the dual and the hidden variable representation of a semiring-based CSP. All the formulations are consistency equivalent.

4.1 Dual Representation

The dual representation of a SCSP problem $P = (S, V, D, C)$ is a problem $P' = (S, V', D', C')$ defined as,

- There is one variable for each constraint in P. We denote x'_c the variable associated to constraint c.
- The domain of x'_c is defined as,

$$D'_c = \{t \in (\prod_{x_i \in var(c)} D_i) \text{ such that } c(t) > \mathbf{0}\}$$

- C' is divided into two groups: unary soft constraints (C'_1) and binary hard constraints (C'_2).

1. There is a unary constraint c' associated to every variable x'_c defined as,

$$c'(t) = c(t) \text{ for all } t \in D'_c$$

2. There is a binary constraint c'_{ij} associated to every pair of variables x'_{c_i} and x'_{c_j} such that $var(c_i) \cap var(c_j) \neq \emptyset$. It is defined as,

$$c'_{ij}(t_i, t_j) = \begin{cases} false, \text{if } [t_i \bowtie t_j = \emptyset] \\ true, \text{otherwise} \end{cases}$$

Theorem:

Let P and P' be a SCSP and its dual. Then, P and P' are consistency equivalent.

Proof:

We show that P and P' are consistency equivalent by defining a bijection between their consistent tuples and showing that it preserves the consistency level of mapped tuples.

Let t be an arbitrary tuple over V. The corresponding tuple over V' is $t' = ((x'_{c_1}, t_1), \ldots, (x'_{c_m}, t_m))$, where $t_i = (t \downarrow_{var(c_i)})$. The consistency level of t in P is,

$$c_P(t) = c_1(t \downarrow_{var(c_1)}) \times \ldots \times c_m(t \downarrow_{var(c_m)})$$

which can be rewritten as,

$$c_P(t) = c_1(t_1) \times \ldots \times c_m(t_m)$$

By definition of P', the consistency level of t' is

$$c_{P'}(t') = \prod_{c'_i \in C'_1} c'_i(t_i) \times \prod_{c'_{ij} \in C'_2} c'_{ij}(t_i, t_j)$$

The join of any pair t_i, t_j is non empty, because both tuples are subtuples of t, therefore $c'_{ij}(t_i, t_j) = \mathbf{1}$. The unit element of the multiplicative operation in the c-semiring is $\mathbf{1}$, therefore the valuation of t' can be rewritten as,

$$c_{P'}(t') = c'_1(t_1) \times \ldots \times c'_m(t_m)$$

By definition of c'_i, this is equal to the consistency level of t. Consequently, t is consistent iff t' is consistent and both tuples have the same level of consistency.

There are tuples over V' that cannot be obtained with the mapping defined above. These are the tuples $t' = ((x'_{c_1}, t_1), \ldots, (x'_{c_m}, t_m))$ having at least one pair of components that do not match (i.e. their join is the empty tuple). However, those tuples are necessarily inconsistent, because the pair of non matching components causes a $\mathbf{0}$ in a binary constraint valuation. Since the multiplicative operation is monotonically decreasing with respect the partial order, the valuation of t' must be $\mathbf{0}$.

4.2 Hidden Variable Representation

The hidden variable representation of a SCSP problem $P = (S, V, D, C)$ is $P' = (S, V', D', C')$ defined as,

- There is one variable for each constraint in P and one constraint for each variable in P. We denote x'_c the variable associated to constraint c and x'_i the variable associated to variable x_i.
- The domain of x'_c is defined as,

$$D'_c = \{t \in (\prod_{x_i \in var(c)} D_i) \text{ such that } c(t) > \mathbf{0}\}$$

- The domain of x'_i is D_i.
- C' is divided into two groups: unary soft constraints (C'_1) and binary hard constraints (C'_2).
  1. There is a unary constraint c' associated to every variable x'_c defined as,
  $$c'(t) = c(t) \text{ for all } t \in D'_c$$
  2. There is a constraint c'_{ij} associated to every pair of variables x'_{c_i} and x'_j such that $x_j \in var(c_i)$. It is defined as,
  $$c'_{ij}(t_i, v) = \begin{cases} true, \text{if}[t_i \downarrow_j = v] \\ false, \text{otherwise} \end{cases}$$

Theorem:

Let P and P' be a SCSP and its hidden variable representation. Then, P and P' are consistency equivalent.

The proof is similar to the previous one and we do not include it here for lack of space.

5 Conclusions

In this paper we have shown that duality results on classical CSPs are preserved in soft constraint CSPs. The definition of the dual and hidden variable representation of SCSP is a direct extension of the method in the classical framework, so the importance of this paper does not relay in its originality. However, our observation indicates the expressiveness power of unary soft constraints. Since solving SCSPs is conceptually more complex than solving classical CSPs, problems with binary hard constraints and unary soft constraints may provide a useful starting point for algorithmic development in the SCSP framework. Some problems that can be directly expressed with binary hard and unary soft constraints are: *weighted independent set*, *weighted clique* [4], *combinatorial auctions* [2] (which is equivalent to *weighted set packing* [4]) and *aircraft landing scheduling* [3].

References

[1] J. Larrosa, P. Meseguer and T. Schiex. Maintaining reversible DAC for Max-CSP. Artificial Intelligence Journal, 107(1).

[2] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. Proceedings of *IJCAI'99*.

[3] J.E. Beasley, M. Krishnamoorthy, Y.M. Sharaiha and D. Abramson. Scheduling aircraft landings - the static case. The Management School, Imperial College. Working paper.

[4] V.T. Paschos. A survey of Approximately Optimal Solutions to Some Covering and Packing Problems. The Management School, Imperial College. Working paper.

[5] M. Jampel A Brief Overview of Over-Constrained Systems. Lecture Notes in Computer Science, vol 1106. 1996.

[6] F. Rossi, C. Petrie and V. Dhar. On the equivalence of constraint satisfaction problems. Proceedings of *ECAI'90*.

[7] R. Dechter and J. Pearl. Tree Clustering for Constraint Networks. *Artificial Intelligence*, 38:353–366, 1989.

[8] R. Dechter. On the expressiveness of networks with hidden variables. Proceedings of *AAAI'90*.

[9] H. Fargier and J. Lang. Uncertainty in constraint satisfaction problems: a probabilistic approach. Proceedings of *ECSQARU*. LNCS 747.

[10] Z. Ruttkay. Fuzzy constraint satisfaction. Proceedings of *IEEE Int. Conf on Fuzzy Systems*. 1994

[11] D. Dubois, H. Fargier and H. Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. Proceedings of *IEEE Int. Conf on Fuzzy Systems*. 1993

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

[13] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie and H. Fargier. Semiring-based CSPs and Valued CSPs: Frameworks, Properties and Comparison. Constraints 4(3), 1999.