The Knowledge Engineering Review, Vol. 31:5, 429–439. © Cambridge University Press, 2016
doi:10.1017/S0269888916000242
First published online 20 December 2016

Revisiting dynamic constraint satisfaction for model-based planning

JEREMY FRANK

NASA Ames Research Center, Moffett Field, CA, USA;
e-mail: jeremy.d.frank@nasa.gov

Abstract

As planning problems become more complex, it is increasingly useful to integrate complex constraints on time and resources into planning models, and use constraint reasoning approaches to help solve the resulting problems. Dynamic constraint satisfaction is a key enabler of automated planning in the presence of such constraints. In this paper, we identify some limitations with the previously developed theories of dynamic constraint satisfaction. We identify a minimum set of elementary transformations from which all other transformations can be constructed. We propose a new classification of dynamic constraint satisfaction transformations based on a formal criteria, namely the change in the fraction of solutions. This criteria can be used to evaluate elementary transformations of a constraint satisfaction problem as well as sequences of transformations. We extend the notion of transformations to include constrained optimization problems. We discuss how this new framework can inform the evolution of planning models, automated planning algorithms, and mixed-initiative planning.

1 Introduction

Automated planning can be posed as a constraint satisfaction problem (CSP), and subsequently solved using CSP techniques (e.g. Do & Kambhampati, 2000; Vidal & Geffner, 2006; Banerjee, 2009). Automated scheduling also makes extensive use of constraint satisfaction techniques (e.g. Laborie, 2003). As planning problems become more complex, it is increasingly useful to integrate constraints on time and resources into planning models, and use complex constraint reasoning approaches to help solve the resulting planning problems (e.g. Ghallab & Laurelle, 1994; Frank *et al*., 2000; Frank & Jónsson, 2003). Similarly, many planning problems are optimization problems; one variation includes plan quality measured by action cost (e.g. Keyder & Geffner, 2008) or the set of satisfied goals (e.g. van den Briel *et al*., 2004). As an intermediate step between satisficing and optimization problems, a problem instance may be modified by the addition or subtraction of goals, or the changing or waiving of some constraints. These incremental modifications of the problem often takes place in a 'mixed-initiative' setting, where human planners use automation to solve problems, as described in Bresina *et al*. (2005).

During the search process, a plan can be translated into an underlying CSP, on which reasoning (propagation and heuristics computations) are performed. During planning, the CSP may be modified by the addition or subtraction of variables, domain values, and constraints. In order to avoid regeneration of the entire CSP multiple times during planning, the underlying CSP machinery can use *dynamic* constraint satisfaction problem (DCSP) techniques. Previous formalisms for dynamic constraint satisfaction have been developed to support automated planners. However, these formalisms are unsatisfactory for several reasons, as described below.

We propose a new formal criteria to classify dynamic constraint satisfaction transformations, and identify a minimum set of transformations from which all other transformations can be constructed.

This criteria can be used to evaluate elementary transformations of a CSP as well as sequences of transformations. We extend the new formalism to include optimization problems, leading to a novel integration of dynamic constraint satisfaction and partial constraint satisfaction. The resulting set of simple transformations allows analysis of every modification of CSPs, with and without optimization criteria. We then discuss how the new framework informs automated planning algorithms, mixed-initiative planning, and the evolution of planning models. This paper summarizes key results from Frank (2014).

2 Dynamic constraint satisfaction: a critical review

The DCSP was originally formalized by Dechter and Dechter (1988). In this formalism, all variables have identical domains, and the set of constraints and variables is allowed to vary over time, thereby creating a sequence of CSPs. Adding variables and constraints are termed *restrictions*, and removing variables and constraints are termed *relaxations*. The focus in this work was on maintaining one or a set of solutions as the problem is changed. A restricted variation of the DCSP was introduced by Mittal and Falkenhainer (1990). In their formalization, the set of variables and constraints is fixed, but the constraints are activated by variable assignments, that is, all of the variables in the scope of a constraint must be activated for the constraint to apply. The set of variables that must be assigned can be a function of other variables' values. There is also a minimum subset condition to ensure no 'extra' variables are assigned. Thus, the problem does not change via some external entity adding variable and constraints, but the act of solving increases the set of required active variables, and thus constraints. The Disjunctive Temporal Network (DTN) introduced by Tsamardinos and Pollack (2003) is focused on a discrete choice of which temporal constraint to apply to a pair of temporal variables. Changing the value of a discrete variable requires removing one temporal constraint and replacing it with another. An extension to Mittal and Falkenhainer's DCSP was introduced by Soininen *et al*. (1999) to allow disjunctive activity constraints (satisfaction of activity constraints implies some subset of variables must be assigned), include 'null' variable assignment, and change the definition of solution to a fixed point instead of minimum subset, leading to reduced computational complexity. An extension to the DCSP to support automated planning was introduced by Jónsson and Frank (2000). In addition to adding or removing variables and restricting or relaxing constraints, new values for existing variables may be added, or values of variables can be removed from the domains permanently. Adding new constraints and changing the scope of constraints was not considered. This approach is elaborated in Frank and Jónsson (2003).

We shall now describe some limitations with these formalisms for DCSPs. Previously, changes to CSPs were simply labeled restrictions (or relaxations) with no formal criteria to determine which is which. Consider adding a variable without constraints to a CSP. Compared with the previous CSP (without the new variable), a decision is now required in the new CSP where no decision was required before. The resulting transformation is labeled a restriction. However, the label is somewhat unintuitive; adding a variable with no constraints is not obviously restrictive, since adding a new variable X with domain size $\geqslant 2$ increases the *number of assignments* and the *number of solutions* but not the *percentage of assignments that are solutions*. In fact, the percentage of assignments that are solutions is left unchanged. That is because the new variable's values are *unconstrained* and therefore increase both the number of assignments and the number of solutions by the same constant factor (namely $|d(X)|$).

The formalisms of DCSPs have not been extended to problems with costs and quality bounds. How is a problem of satisfiability transformed into an optimization problem? Does introducing optimization restrict or relax the problem? Suppose the problem is an optimization problem, and initially solutions below some cost are demanded. Over time, the cost bound is found to be too low, and the problem is changed by either increasing the cost bound, or by reducing (or eliminating) the cost of assignments associated with some constraints. Intuitively, making the cost bound higher might seem like a relaxation. Similarly, making some solutions have higher cost might seem like a restriction. If we are in an optimal setting with a cost bound defining the set of feasible solutions, then increasing the bound will increase the number of solutions, and decreasing cost will only reduce the number of solutions. However, if costs on single assignments or constraints are changed, but the cost bound defining solutions remains the same, it is not obvious *a priori* how many assignments' costs will remain below the cost bound, and therefore whether the change is a restriction or a relaxation.

Finally, consider transforming a problem in which constraints cannot be waived into one where they can. This is a significant transformation; a new variable must be introduced, the scope of the waivable constraint must be changed, and the relations in the constraint extended. While the change is obviously a relaxation, existing DCSP formalisms lack good support for this change. In Jónsson and Frank (2000) such a change is not considered; while adding constraints was conceived of in Dechter and Dechter (1988), a change of scope was not, and must be synthesized from other primitives in the other formalisms. Other changes may not be so easily analyzed.

3 Dynamic constraint satisfaction: a new formalism

Throughout the remainder of this paper, we assume CSPs variables all have finite discrete domains.

Definition 1. *Let $X_1, \ldots, X_n$ be a set of variables. The domain of variable X_i is denoted $d(X_i)$. Let $x_{i_j} \in d(X_i)$ be a value. A Constraint C_j is a tuple S_j, R_j. The scope S_j of the constraint is a set of variables $X_{j1}, \ldots, X_{jk}$. The relation $R_j \subseteq d(X_{j1}) \times \ldots \times d(X_{jk})$ is a list of tuples r_{jh} defining the allowed combinations of values to the variables in the scope of the constraint. A constraint satisfaction problem or CSP $\mathcal{P}$ is a set of variables $X_1, \ldots, X_n$ with domains $d(X_1) \ldots, d(X_n)$ and a set of constraints $C_1, \ldots, C_m$. The projection of an assignment $x \in d(X_1) \times \ldots \times d(X_n)$ onto a set of variables X, denoted $\pi(x, X)$, is the value of each variable in X. An assignment is a solution if its projection onto the scope of each constraint C_i is in the relation R_i, that is, if $\forall C_i\, \pi(x, S_i) \in R_i$.*

We now define DCSPs using the notion of transformations between CSPs.

Definition 2. *A transformation τ is a function that maps a CSP P_i to a CSP P_j denoted $\mathcal{P}_i \xrightarrow{\tau} \mathcal{P}_j$. Let n be the number of variables in $\mathcal{P}_i$. Let m be the number of constraints in $\mathcal{P}_i$. Let $d_i = |d(X_i)|$. Let $c_j = |R_j|$. The classes of transformations are:*

1. *Add X_{n+1} to $\mathcal{P}_i$ with $|d(X_{n+1})| = 1$. Denote this transformation $X+$.*
2. *Remove X_j with $|d(X_j)| = 1$ s.t. $\forall C_k\, X_j \notin S_k$ from $\mathcal{P}_i$. Denote this transformation $X-$.*
3. *Add a unique value $x_{i_{d_i+1}}$ to $d(X_i)$ s.t. $\forall C_k\, X_i \notin S_k$. Denote this transformation $d+$.*
4. *Remove a value x_{i_j} from $d(X_i)$ s.t. $\forall C_k$, $X_i \notin S_k$. Denote this transformation $d-$.*
5. *Add a unique tuple $r_{j_{c_j+1}}$ to R_j. Denote this transformation $r+$.*
6. *Remove a tuple r_{jh} from R_j. Denote this transformation $r-$.*
7. *Add C_{m+1} with $S_{m+1} = X_{m+1_1} \ldots X_{m+1_k}$ and relation $R_{m+1} = \times_{i=1}^{k} d(X_{m+1_i})$ to $\mathcal{P}_i$. Denote this transformation $C+$.*
8. *Remove C_j with $R_j = \times_{i=1}^{k} d(X_{jk})$ from $\mathcal{P}_i$. Denote this transformation $C-$.*

In the spirit of Dechter and Dechter (1988) and Jónsson and Frank (2000), a DCSP allows an external entity to apply a (possibly infinite) sequence of the above transformations to a CSP, and request a solution after any subsequence of transformations. An example is shown in Figure 1.

Definition 3. *Let $L(\mathcal{P})$ be the number of solutions to $\mathcal{P}$. The fraction of solutions denoted $L_p(\mathcal{P})$ is then $\frac{L(\mathcal{P})}{\prod_{i=1}^{n} |d(X_i)|}$. Let τ be a transformation from $P_i \xrightarrow{\tau} P_j$. A relaxation increases the fraction of solutions, that is, $L_p(\mathcal{P}_i) < L_p(\mathcal{P}_j)$; a restriction decreases the fraction of solutions, that is, $L_p(\mathcal{P}_i) > L_p(\mathcal{P}_j)$. A neutral transformation is neither a restriction or a relaxation.*

Unlike previous theories of transformations on DCSPs, the classes transformations on DCSPs cannot all be classified as restrictions, relaxations, or neutral:

Theorem 1. *There are classes of transformations that are restrictions, relaxations, and neutral. Furthermore, for a DCSP P_i, there are classes of transformations that can be either restrictions or neutral, and there are classes of transformations that can be relaxations or neutral.*

The proof employs straightforward analysis of each class of transformation:

1. $X+$: since $|d(X_{n+1})| = 1$, $L(\mathcal{P}_j) = (L(\mathcal{P}_i))(|d(X_{n+1})|) = L(\mathcal{P}_i)$. Similarly, $\left(\prod_{i=1}^{n} |d(X_i)|\right)(|d(X_{n+1})|) = \prod_{i=1}^{n} |d(X_i)|$. Adding a variable with a single value but not adding this value to any relations leaves the

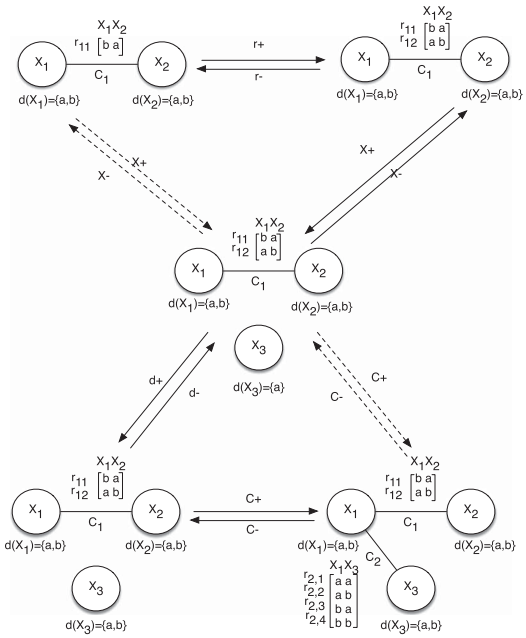

Figure 1 Transformations. The set of all valid dynamic constraint satisfaction problem (DCSP) transformations. The dotted lines show valid transformations, but not the exact transformations of the DCSP in the figure

number of assignments and solutions unchanged, so the fraction of solutions is unchanged, hence this is a neutral transformation.

2. $X-$: removing a variable with a single value that participates in no constraints leaves the number of assignments and solutions unchanged, so the fraction of solutions is unchanged, hence this is a neutral transformation.
3. $d+$: we show that adding a value to a variable but leaving the relations unchanged increases the number of assignments and the number of solutions by the same factor. Let $d+$ change $d(X_n)$ w.l.o.g. (simplifies notation for the computation). Then the number of assignments increases by a factor of

$$\frac{\left(\prod_{i=1}^{n-1} |d(X_i)|\right)((|d(X_n)|+1))}{\left(\prod_{i=1}^{n-1} |d(X_i)|\right)(|d(X_n)|)} = \frac{|d(X_n)|+1}{|d(X_n)|}$$

Next, recall values can only be added to variables not in the scope of any constraint. Let $\mathcal{Q}_i$ be $\mathcal{P}_i - X_n$, that is, the CSP without variable X_n. Rewrite $L(\mathcal{P}_i) = (L(\mathcal{Q}_i))(|d(X_n)|)$. So the number of solutions increases by a factor of

$$\frac{(L(\mathcal{Q}_i))(|d(X_n)+1|)}{(L(\mathcal{Q}_i))(|d(X_n)|)} = \frac{|d(X_n)|+1}{|d(X_n)|}$$

which means the increase in the number of solutions and the number of assignments increase by the same factor, so the fraction of solutions remains unchanged.

4. $d-$: removing a value from a variable domain leaves the fraction of solutions unchanged as argued above.
5. $r+$: adding a tuple to a relation may not increase the number of solutions, as the tuple may be excluded by other relations. However, adding a unique tuple to a relation cannot decrease the number of solutions. The number of assignments does not change, so the fraction of solutions cannot decrease; hence this is either a neutral transformation or a relaxation.

6. $r-$: removing a unique tuple from a relation cannot increase the number of solutions. The number of assignments does not change, so the fraction of solutions cannot increase, hence this is a neutral transformation or a restriction.
7. $C+$: adding a constraint with the relation consisting of all assignments to the variables in the scope leaves the fraction of solutions unchanged, hence this is a neutral transformation.
8. $C-$: removing a constraint whose relation consists of all assignments to the variables in the scope leaves the fraction of solutions unchanged, hence this is a neutral transformation.

Definition 3 can now be used to classify a single transformation, and also to evaluate a sequence of transformations. We now have a principled definition of restriction and relaxation in terms of the fraction of solutions. We also have finer-grained and more precisely characterized transformations than previous formalisms. We see that transformations need not be restrictions or relaxations, and that some transformations previously identified as restrictions or relaxations are classified differently using our new definition. For example, adding a variable is considered a *restriction* in Dechter and Dechter (1988) but is neither a restriction nor a relaxation according to the new classification.

Theorem 1 holds regardless of whether the definition of restriction or relaxation uses the number or fraction of solutions. This is summarized in the table below. Recall from Definition 3 that $L(\mathcal{P})$ is the number of solutions, and $L_p(\mathcal{P})$ is the fraction of solutions.

τ	$\Delta L(\mathcal{P})$	$\Delta \prod_{i=1}^{n} \lvert d(X_i)\rvert$	$\Delta L_p(\mathcal{P})$
$X+$	0	0	0
$X-$	0	0	0
$d+$	$x>0$	$x>0$	0
$d-$	$y<0$	$y<0$	0
$C+$	0	0	0
$C-$	0	0	0
$r+$	$\geqslant 0$	0	$\geqslant 0$
$r-$	$\leqslant 0$	0	$\leqslant 0$

The table shows that only a small number of transformations actually change the set of solutions in a meaningful way. Consider the transformation $\tau = d+$ (i.e. adds values to $d(X_i)$). Since X_i cannot be involved in any constraints, no relations in existing constraints are modified, and no constraints are added, all of the new values participate in solutions to $\mathcal{P}_j$. However, every solution to $\mathcal{P}_i$ is a solution to $\mathcal{P}_j$. In a sense, the transformation is *trivial*, in that little work is required to keep up with this 'restriction'. Similarly, the transformation $d-$ (removing a value) modifies a variable that is not in any relations, and thus reduces the number of assignments, and the solutions are reduced by the same fraction. The only non-trivial transformations are those in which the set of tuples in an existing relation are modified.

It is possible to synthesize the previously defined restrictions and relaxations from a sequence of these new, primitive transformations. We provide a specific example in Figure 2 of adding a variable to the scope of a constraint to support waiving the original constraint. In this figure some of the transformations have been combined for brevity. This transformation is contrasted to the transformation of simply removing a constraint from the variables. The complex sequence of removing the old constraint and replacing it in this case could be made slightly easier via a neutral transformation $+S$ that increases the scope of a variable and extends the old relations with new ones allowing every value of the newly in-scope variable; similarly, the transformation $-S$ is permitted to remove the scope of any variable all of whose values are allowed in the relation on the variables that will remain in scope.

4 Extending dynamic constraint satisfaction problems to optimization problems

We now show how to extend the notion of DCSPs to optimization problems; more precisely, we address problems in which assignments have costs, and there is a cost bound defining the set of feasible solutions. We choose the partial constraint satisfaction problems (PCSPs) formalism to extend the analysis of

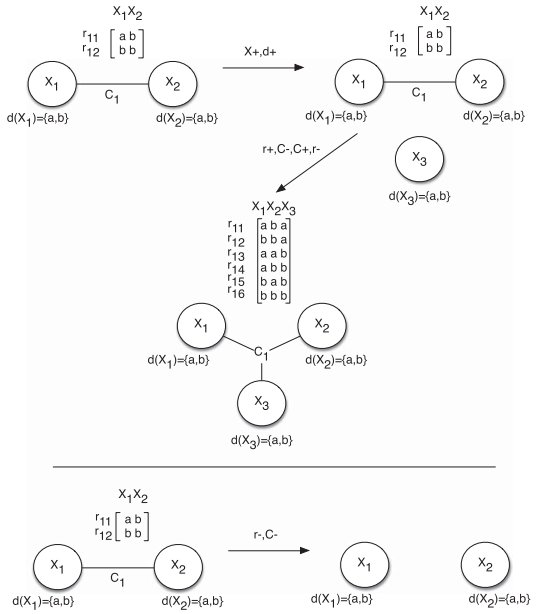

Figure 2 Extending a constraint to allow waiving violations by variable assignment during solving (top) versus removing a constraint (bottom)

dynamic constraint satisfaction. PCSPs were originally defined by Freuder and Wallace (1992). While more modern formalisms such as soft constraints defined over semirings (Bistarelli *et al.*, 1995) have more expressive power and more sophisticated theoretical grounding, we will develop the theory of DCSPs using PCSPs for simplicity. PCSPs extend CSPs by adding a cost function that maps a relation to a real value, and the goal is to find an assignment of values such that the sum of costs does not exceed a bount B:

DEFINITION 4. *A partial constraint C_j is a tuple S_j, R_j, f_j. The scope S_j of the constraint is a set of variables $X_{j1}, \ldots, X_{jk}$. The relation $R_j \subseteq d(X_{j1}) \times \ldots \times d(X_{jk})$. Finally, $f: R_j \rightarrow R_{\geq 0}$. A partial constraint satisfaction problem or PCSP $\mathcal{P}$ is a set of variables $X_1, \ldots, X_n$ and domains $d(X_1), \ldots, d(X_n)$ and a set of partial constraints $C_1, \ldots, C_m$ and a real number B. A solution is an assignment x such that* $\sum_{r_{jh} \mid \exists C_j \pi(x, S_j) = r_{jh}} f(r_{jh}) < B$.

We will refer to f as the cost of a relation, since solutions must have a cumulative value below B. This definition allows the cost function of PCSPs with partially defined relations (i.e. f defined on a subset of $d(X_{j1}) \times \ldots \times d(X_{jk})$) to be well defined; specifically, an assignment x with a projection $\pi(x, S_j)$ not equal to any r_{jh} contributes nothing to the cost of the assignment. This 'inverts' the semantics of relations in the DCSP, in which the satisfying relations are enumerated; now the relations imposing a 'cost' are enumerated, and only these relations may eliminate solutions. The PCSP formulation generalizes naturally to constrained optimization problems, in which an assignment minimizing the cost is desired, but solutions exceeding the cost are forbidden. Alternatively, if B is arbitrarily large and minimum cost solutions are desired, the resulting problem is similar to the MAX-CSP (Wallace, 1996).

In order to extend DCSPs to optimization problems, we must add some new transformations. First, we must be able to change the cost of any tuple in a PCSP:

DEFINITION 5. *Denote the transformation that increases the value of tuple $r_{j_h} \in C_j$ by $f+$.*
Denote the transformation that decreases the value of tuple $r_{j_h} \in C_j$ by $f-$.

Strictly speaking, adding and removing constraints and tuples from relations are different for CSPs and PCSPs. In order to avoid confusion we add new notation for these transformations when applied to PCSPs.

DEFINITION 6. *Denote the transformation that adds C_{m+1} with $S_{m+1} = X_{m+1_1} \ldots X_{m+1_k}$ and relation $R_{m+1} = \varnothing$ by $B+$.*

Denote the transformation that removes C_j with $S_j = X_{j1}, \ldots, X_{jk}$ and relation $R_j = \varnothing$ by $B-$.
Denote the transformation that adds a tuple r_{j_h} to R_j with $f\left(r_{j_h}\right) = 0$ by $s+$.
Denote the transformation that removes a tuple r_{j_h} from R_j with $f\left(r_{j_h}\right) = 0$ by $s-$.

The transformations $X+$, $X-$, $d+$, $d-$ are identical to their DCSP counterparts.

We now show how a CSP $\mathcal{P}_i$ can be transformed into a PCSP $\mathcal{P}_j$. Doing so requires transforming C_j, or more specifically, $R_j \in \mathcal{P}_i$, into $(R_j^{'}, f_j) \in \mathcal{P}_j$. For each $r_{j_h} \in d\left(X_{j_1}\right) \times \ldots \times d\left(X_{j_k}\right)$ such that $r_{j_h} \notin R_j$, add r_{j_h} with $f\left(r_{j_h}\right) = 1$ to $R_j^{'}$. Let $B = 1$. Let x be a satisfying assignment to $\mathcal{P}_i$. Then $\pi(x, S_i) \notin R_j^{'}$, and therefore contributes 0 to the sum $\sum_{r_{j_h} | \exists C_j \pi\left(x, S_j\right) = r_{jh}} f\left(r_{j_h}\right)$ by definition. For each solution x to the original CSP, the sum is therefore below the bound 1. Any other assignment has a sum of at least 1 and therefore does not satisfy the inequality. We can then replace R_j with $R_j^{'}$ as the relation for C_j in the new PCSP. We call this new transformation $v+$. Similarly, we can define the inverse transformation $v-$ which can only be performed if $\forall C_j \forall r_{j_h} \in C_j f\left(r_{j_h}\right) = 1$ and $B = 1$. These transformations are neutral, in the sense that all satisfying assignments continue to satisfy. The transformation is polynomial time in the size of the (P)CSP. As noted above, the transformation 'inverts' the set of tuples in the relations, since we add the tuples that will exceed the cost bound and therefore 'violate' the constraints.

DEFINITION 7. *Denote the transformation from a CSP into PCSP by $v+$.*
Denote the transformation from a PCSP into CSP by $v-$.

Note: while the presence of elements of the relations with $f\left(r_{j_h}\right) = 0$ may seem unnecessary (as well as a waste of space) in a PCSP, these relations are made explicit to simplify the description of the transformation $s+$ followed by $f+$, or $f-$ followed by $s-$.

Finally, consider a transformation to increase or decrease B. This transformation arises naturally in branch-and-bound search, and may also arise during modeling. Is this equivalent to a series of transformations $+f$, $-f$? Trivially, the answer is no. Making a single change $+f$, $-f$ can introduce a small change to the solutions that is impossible to mimic with a change to B, as shown in Figure 3.

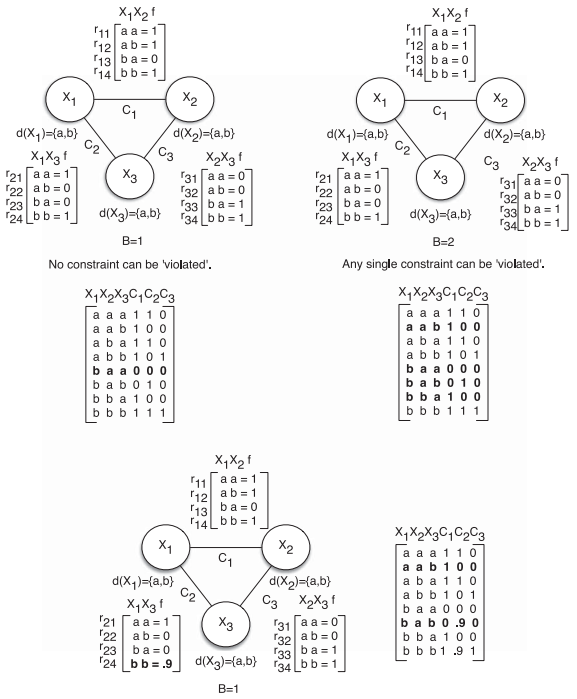

Figure 3 Changes to solutions via $f+$,$f-$ may not be achievable via a change to B. This partial constraint satisfaction problem (top left) is a transformed constraint satisfaction problem allowing no constraint 'violations' (i.e. the only solution has accumulated cost 0). Changing B from 1 to 2 (top right) allows one 'violation', thereby introducing three new solutions. This is the minimum number of new solutions that can be achieved. By contrast, if $f(r_{2,\,4})$ of constraint C_2 is changed from 1 to 0.9 (bottom) then one new solution is introduced

By contrast, can changes to the solutions due to changes to B be accomplished by a set of changes $+f$, $-f$? The trivial answer is yes.

THEOREM 2 *Let $\mathcal{P}_i \xrightarrow{\tau} \mathcal{P}_j$ transform a PCSP by means of a change in bound from B to $B+b$ or $B-b$. Then there is a series of transforms $\mathcal{P}_i \xrightarrow{f+,f-} \mathcal{P}_k$ such that $L(\mathcal{P}_j) = L(\mathcal{P}_k)$.*

Recall the criteria for x to be a solution to a PCSP is $\sum_{r_{j_h} | \exists C_j \pi(x, S_j) = r_{j_h}} f(r_{j_h}) < B$. If we transform B to $B+b$ to admit more solutions, we can instead scale each $f(r_{j_h})$ by a factor of $\frac{B}{B+b}$ and achieve an identical result. To show the cost of the solutions after these transformations satisfy the old bound:

$$\begin{aligned}
&\sum_{r_{j_h} | \exists C_j \pi(x, S_j) = r_{j_h}} \left(\frac{B}{B+b}\right) f(r_{j_h}) \\
&= \left(\frac{B}{B+b}\right) \left(\sum_{r_{j_h} | \exists C_j \pi(x, S_j) = r_{j_h}} f(r_{j_h}) \right) \\
&< \left(\frac{B}{B+b}\right)(B+b) = B
\end{aligned}$$

The same argument holds for assignments that are not solutions, and if we reduce B. Notice that the ranking of optimal solutions is also preserved.

While transforming the bounds can, in fact, be simulated by transformations $f+, f-$, the fact that the required transformations impact *every* tuple in *every* constraint makes for a somewhat indiscriminate transformation. This justifies the inclusion of more concise bounds changes in our list of transformations.

DEFINITION 8. *Denote the transformation that raises the bound by $b+$.*
Denote this transformation that lowers the bound by $b-$.

In Figure 4 we show the transformations from CSPs to PCSPs, and between PCSPs.

THEOREM 3 *There are classes of transformations on PCSPs that are restrictions, relaxations, and neither restrictions nor relaxations. Furthermore, for a PCSP $\mathcal{P}_i$, there are classes of transformations that can be either restrictions or neutral, and there are classes of transformations that can be relaxations or neutral.*

Increasing the cost of a tuple $f+$ or decreasing the bound $b-$ may not decrease the number of solutions or fraction of solutions, but they cannot increase the number or fraction of solutions. Vice versa, decreasing the cost of a tuple $f-$ or increasing the bound $b+$ may not increase the number of solutions or fraction of solutions, but they cannot decrease the number or fraction of solutions. The transformations for PCSPs are shown in the table below.

τ	$\Delta L(\mathcal{P})$	$\Delta \prod_{i=1}^{n} \lvert d(X_i) \rvert$	$\Delta L_p(\mathcal{P})$
$X+$	0	0	0
$X-$	0	0	0
$d+$	$x>0$	$x>0$	0
$d-$	$y<0$	$y<0$	0
$B+$	0	0	0
$B-$	0	0	0
$s+$	$\geqslant 0$	0	$\geqslant 0$
$s-$	$\leqslant 0$	0	$\leqslant 0$
$f+$	$\leqslant 0$	0	$\leqslant 0$
$f-$	$\geqslant 0$	0	$\geqslant 0$
$b-$	$\leqslant 0$	0	$\leqslant 0$
$b+$	$\geqslant 0$	0	$\geqslant 0$
$v+$	0	0	0
$v-$	0	0	0

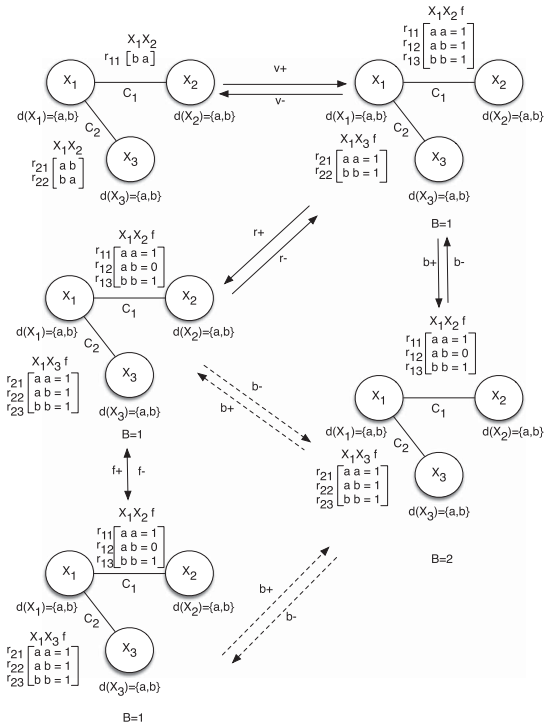

Figure 4 Transformations between constraint satisfaction problems and partial constraint satisfaction problems. Set of all valid transformations. The dotted one shows a valid transform but not the exact evolution of the dynamic constraint satisfaction problem in the figure

5 Planning using dynamic constraint satisfaction problems: discussion

As described in the introduction, some planners already use a DCSP framework during search. In this section, we discuss how this new formalization of DCSPs could be used for planning, both as a foundation for search and also during planning model development.

Recall that a Simple Temporal Constraint has the form $a \leqslant |X_i - X_j| \leqslant b$. A Disjunctive Temporal Constraint has scope X_i, X_j, X_k; $d(X_k)$ is a finite discrete set, and $d(X_i) = d(X_j) = Z$. For each $x_k \in d(X_k)$ there is a pair of constants a_x, b_x; the constraint has the form $(X_k = x_k) \Rightarrow (a_x \leqslant |X_i - X_j| \leqslant b_x)$. What happens if a new assignment $X_k = x_k$ is made in a DTN during search after determining that the previous value y_k resulted in a temporal constraint violation? One approach is to have the planner maintain the mapping between discrete values of the variable X_k and a Simple Temporal Network by transforms $C+$, $C-$ executed during search. When a violation is detected during search and assignment $X_k = y_k$ is revoked, the violated Simple Temporal Constraint $(a_y \leqslant |X_i - X_j| \leqslant b_y)$ is removed, as described in Figure 2 (bottom); when assignment $X_k = x_k$ is made, the new constraint $(a_x \leqslant |X_i - X_j| \leqslant b_x)$ is added. This approach is used in several planners (e.g. Ghallab & Laurelle, 1994; Frank & Jónsson, 2003); by contrast, the approach in Do and Kambhampati (2000) is to construct the CSP incrementally and extract actions from the solution. The potential advantage of the new approach is early detection of no-goods as relations are added to constraints. However, this comes at the price of adding individual relations during search, which is impractical for large discrete constraints, and impossible for continuous constraints.

A second potential use is in the incremental modeling of a planning domain. Simple planning problems (linear STRIPS planning in which all preconditions are positive) do not require sophisticated analysis using constraints; adding preconditions and negative effects cannot increase the number of plans, and adding positive effects cannot decrease the number of plans. However, more complex planning and scheduling problems (including negative preconditions and partial orders in STRIPS planning) may benefit from this new framework at modeling time. A recent line of research known as Model-Lite Planning due to Kambhampati (2007) contemplates early-stage models for criticism as opposed to

planning. A formalism such as the one proposed here could be used for such a purpose. Suppose new conditions or effects are added to some activity as part of the modeling process. Is the change a restriction or relaxation? Is it easy to tell immediately? If the change is not obvious, or perhaps even if it is, informing the modeler of the consequences of such a change at modeling time may be valuable. In Frank (2014) an analysis of changes to STRIPS models and the resulting CSP using the approach in Do and Kambhampati (2000) reveals that some domain model changes can lead to multiple restrictions and relaxations of the resulting CSP, making it hard to know if plans are being added or removed. Specifically, adding effects to actions can lead to additional constraints (e.g. static mutexes) but also eliminate constraints (establishing conditions not true in the initial state).

A third use of the new formalism is to analyze the transformation of planning as satisfiability into an optimization problem at modeling time, for example, using the *v+* transformation. Early in model development it may not be apparent whether the right problem is a satisficing or optimization problem; a 'push-button' option to change the problem offers significant advantages. Consider, for instance, integrating a simple report on the consequences of adding or removing conditions and effects to a tool such as itSimple, described in Vaquero *et al*. (2007), for use in a plan or model critique phase of Model-Lite planning. Feedback for simple situations like adding or removing conditions and effects, as analyzed in the earlier version of this paper (Frank, 2014), may provide value, as would analysis when satisficing problems are transformed into optimization problems. However, successful application of this idea is complicated by two factors. The first, and more minor problem, is that, as described, the PCSP framework proposed in this paper uses a cost upper bound; while some planning problems are naturally posed in terms of action cost (e.g. Keyder & Geffner, 2008), others may be better formalized as problems of maximizing the benefit of achieved goals (e.g. van den Briel *et al*., 2004). Addressing problems of maximizing benefit requires a minor change in the PCSP framework. The second, more difficult problem, is to find a suitable DCSP representation of the planning problem for representing plans with costs. For instance, while it might be possible to create a CSP formulation with costs incurred if goals are not achieved, it may be hard to do so for makespan minimization objectives.

6 Conclusions and future work

In this paper, we present a new classification of dynamic constraint satisfaction transformations based on a quantifiable criteria: the change in the fraction of solutions $\Delta L_p(\mathcal{P})$. This criteria can be used to evaluate elementary transformations of a CSP as well as sequences of transformations. We identify a core set of transformations from which all other transformations can be constructed. We extend the notion of transformations to include optimization problems. The resulting transformations are shown to consist of restrictions, relaxations, and neutral transformations. For optimization problems, classes of transformations may contain more than one type of transformation. We identify a complete set of transformations that can transform a problem from a satisficing problem to an optimization problem, or back.

There are a variety of avenues for future work in this area. Principle among them is extending this framework to richer constraint satisfaction formalisms. While the analysis works for finite domain constraints where enumeration of solutions is straightforward, extension to constraints over integers or reals is a new challenge. In addition, while CSPs should be transformable to more sophisticated soft constraints frameworks, for example, semirings, there may be surprises here, and a thorough analysis should be performed.

The analysis of DCSP transformations using the new framework contains no real surprises when compared with previous CSP theories. The most complex transformation, adding or removing the satisfying tuples in a relation, have the most impact on the solutions, but are the hardest to analyze. The most interesting question from the point of view of computational complexity is whether or not deconstructing a transformation into its primitive parts helps 'eager' propagation. Unfortunately there are no easy answers to this question.

The impact of the new formalism on planning is an open question. The most interesting potential use is for support during model development, by identifying model changes that add or remove plans. However, numerous design questions must be answered to apply the framework. How would such information be provided to a modeler? What can modelers do with this information when it is provided? Different

planning to CSP encodings, such as those described in Banerjee (2009) or van den Briel *et al*. (2005), may lead to different results than those described in the earlier version of this paper (Frank, 2014). Extending the analysis for modeling to include more complex formalisms such as time, resources, domain axioms, disjunctive preconditions and conditional effects will extend the power of the approach.

Acknowledgements

The author would like to thank the anonymous reviewers of the paper for their feedback and editorial comments.

References

Banerjee, D. 2009. Integrating planning and scheduling in a CP framework: a transition-based approach. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling*, 330–333.

Bistarelli, S., Montanari, U. & Rossi, F. 1995. Constraint solving over semirings. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 624–630.

Bresina, J., Jónsson, A., Morris, P. & Rajan, K. 2005. Activity planning for the Mars exploration rovers. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling*, 40–49.

Dechter, R. & Dechter, A. 1988. Belief maintenance in dynamic constraint networks. In *Proceedings of the 7th National Conference on Artificial Intelligence*, 37–42.

Do, M. & Kambhampati, S. 2000. Solving planning-graph by compiling it into CSP. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems*, 82–91.

Frank, J. 2014. Revisiting dynamic constraint satisfaction for automated planning. In *Workshop on Constraints and Planning Systems, in conjunction with the 24th International Conference on Automated Planning*.

Frank, J. & Jónsson, A. 2003. Constraint based attribute and interval planning. *Journal of Constraints Special Issue on Constraints and Planning* **8**, 339–364.

Frank, J., Jónsson, A. & Morris, P. 2000. On reformulating planning as dynamic constraint satisfaction. In *Proceedings of the 4th Symposium on Abstraction, Reformulation and Approximation*.

Freuder, E. & Wallace, R. 1992. Partial constraint satisfaction. *Artificial Intelligence* **58**, 21–70.

Ghallab, M. & Laurelle, H. 1994. Representation and control in IxTeT, a temporal planner. In *Proceedings of the 4th International Conference on AI Planning and Scheduling*, 61–67.

Jónsson, A. & Frank, J. 2000. A framework for dynamic constraint reasoning using procedural constraints. In *Proceedings of the 10th European Conference on Artificial Intelligence*, 93–97.

Kambhampati, S. 2007. Model-lite planning for the web-age masses: the challenges of planning with incomplete and evolving domain models. In *Proceedings of the 13th National Conference on Artificial Intelligence*, 1601–1604.

Keyder, E. & Geffner, H. 2008. Heuristics for planning with action costs, revisited. In *Proceedings of the 18th European Conference on Artificial Intelligence*, 140–149.

Laborie, P. 2003. Algorithms for propagating resource constraints in AI planning and scheduling: existing approaches and new results. *Artificial Intelligence* **143**, 151–188.

Mittal, S. & Falkenhainer, B. 1990. Dynamic constraint satisfaction problems. In *Proceedings of the 9th National Conference on Artificial Intelligence*, 25–32.

Soininen, T., Gelle, E. & Niemela, I. 1999. A fixpoint definition of dynamic constraint satisfaction. In *Proceedings of the 5th International Conference on the Principles and Practices of Constraint Programming*, 419–433.

Tsamardinos, I. & Pollack, M. 2003. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence* **151**(1–2), 43–90.

van den Briel, M., Vossen, T. & Kambhampati, S. 2005. Reviving integer programming for AI planning: a branch and cut framework. *Proceedings of the 15th International Conference on Automated Planning and Scheduling*, 562–569.

van den Briel, M., Sanchez Nigenda, R., Do, M. & Kambhampati, S. 2004. Effective approaches for partial satisfaction (oversubscription) planning. In *Proceedings of the 19th National Conference on Artificial Intelligence*.

Vaquero, T., Romero, V., Tonidanel, F. & Silva, J. 2007. ItSimple 2.0: an integrated tool for designing planning domains. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling*, 336–343.

Vidal, V. & Geffner, H. 2006. Branching and pruning: an optimal POCL planner based on constraint programming. *Artificial Intelligence* **170**(3), 298–335.

Wallace, R. 1996. Enhancement of branch and bound methods for the maximal constraint satisfaction problem. In *Proceedings of the 13th National Conference on Artificial Intelligence*, 188–195.