Extending Dynamic Backtracking to Solve Weighted Conditional CSPs

Robert Effinger and Brian Williams

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology

Problem Statement

- Many planning and design problems can be characterized as optimal search over a constrained network of conditional choices with preferences.

- To draw upon the advanced methods of constraint satisfaction to solve these types of problems, many dynamic and flexible CSP variants have been proposed.

- One such variant, the Weighted Conditional CSP (WCCSP), employs activity constraints and soft constraints to model both conditional dependencies and preferences within a unified framework. [1,2]

- So far, however, little work has been done to extend the full suite of CSP search algorithms to solve these CSP variants.

- In this paper, we extend Dynamic Backtracking [3] and similar backjumping-based CSP search algorithms to solve WCCSPs by utilizing activity constraints and soft constraints in order to quickly prune infeasible and suboptimal regions of the search space.

Weighted Conditional CSP (WCCSP)

A WCCSP is a tuple, $\langle I, V, I_I, C_C, C_A, C_S, f(P)\rangle$. Where,

- $I = \{i_1, i_2, \mathrm{K}, i_n\}$, is a set of finite domain variables.
- $V = \{V_1, V_2, \mathrm{K}, V_n\}$, are finite domains for each $i \in I$.
- $I_I \subseteq I$, is a set of initially active variables.
- C_C, is a set of hard constraints that must be satisfied.
- C_A, is a set of activity constraints describing the conditions under which each variable becomes active.
- C_S, is a set of soft constraints of the form $\langle c, f_s(c)\rangle$.
- Where, c is an assignment of values to variables, and $f_s(c) \rightarrow \Re$ is a cost to be incurred if $c \in P$, where P is the current partial solution (Definition 1).
- $f(P)$, assigns a real-valued cost to a partial solution, P, by summing the costs of each soft constraint which is violated by that partial solution, $f(P) = \sum_{c \in P} f_s(c)$.
- An *activity constraint* is an expression of the form $AC \rightarrow active(i_k)$, where AC represents an assignment of values to variables, $\{i_1 = v_1, \mathrm{K}, i_j = v_j\}$, and is the condition under which variable i_k becomes *active*.

Approach

To extend Dynamic Backtracking to solve Weighted Conditional CSPs, we augment the algorithm to appropriately handle activity constraints and soft constraints. This is accomplished via four extensions to the Dynamic Backtracking algorithm:

1.) A total variable ordering, I_O, for searching over conditional variables, and a conditional variable instantiation function.

2.) A modified backjumping resolution step which accounts for the conditional variables.

3.) A recursive check to remove deactivated variables from the partial solution when backjumping.

4.) A branch-and-bound search framework augmented to construct minimal suboptimal nogoods.

Pseudocode

Dynamic Backtracking for the WCCSP:

1. *Set $P = \varnothing$, $I = I_I$. Set $E_i = \varnothing$ for each $i \in I$.* **(1)** *Take as input the total variable ordering, I_O.* **(4)** *Set the incumbent solution $N = (\varnothing, \infty)$.*

2.a. **(4)** *If $\hat{P} = I_A$, and $f(P) < f(N)$, P is the new incumbent solution. Set $N = (P, f(P))$.*

2.b. **(4)** *If $\hat{P} = I_A$, set $E_i = E_i \cup (i \neq v, \hat{P} - i)$. Otherwise, select a variable* **(1)** *$i = applyNextAC(\ )$ (Function 1) and set $E_i = E_i \cup$* **(4)** *$\varepsilon_O(P, i)$. (Definition 8)*

3. *Set $L = V_i - \hat{E}_i$. If J is nonempty, choose an element $v \in L$. Add (i, v) to P and return to step 2.*

4. *If L is empty, we must have $\hat{E}_i = V_i$; let E be the set of all variables appearing in the explanations, T, of each elimination explanation, $(i \neq v, T)$ for each $v \in \hat{E}_i$,* **(2)** *plus all of the variables appearing in variable i's activating constraint, AC. (Proposition 4.1, OCCSP Backjumping Resolution Step)*

5. *If $E = \varnothing$,* **(4)** *return the incumbent, N. Otherwise, let (j, v_j) be the last entry in P such that $j \in E$. Remove (j, v_j) from P and for each variable $k \in P$ which was assigned a value after j, remove from E_k any eliminating explanation that involves j.* **(3)** *Call removeUnsupportedVars(j , P), and set,*

$$E_j = E_j \cup \text{(4)}\ \varepsilon_O(P, j) \cup \{(j \neq v_j, E \cap \hat{P})\}$$

so that v_j is eliminated as a value for j because of the values taken by variables in $E \cap \hat{P}$. Now set $i = j$ and return to step 3.

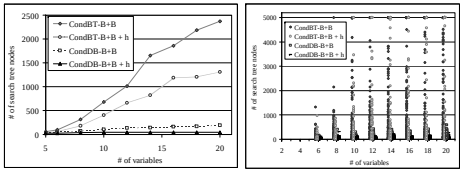
(1) Extension #1, (2) Extension #2, (3) Extension #3, (4) Extension #4

WCCSP Backjumping Resolution Step:

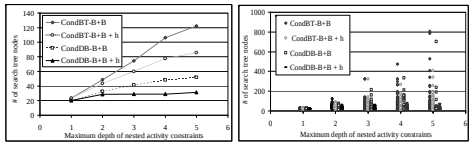
Let i be a variable with domain, $V_i = \{v_1, v_2, \mathrm{K}, v_m\}$, activity constraint $AC \rightarrow active(i)$, and let $P_1, P_2, \mathrm{K}, P_m$ be partial solutions that do not include i. If, $P_1 \cup \{(i, v_1)\}, P_2 \cup \{(i, v_2)\}, \mathrm{K}, P_m \cup \{(i, v_m)\}$ are all nogoods, then, $P_1 \cup P_2 \cup \mathrm{K} \cup P_d \cup AC$ is also a nogood. Note that the new nogood can be resolved by removing variable i from the problem via conceding any one of its activation conditions, AC.

Results

We benchmarked the CondDB-B+B algorithm against a standard branch-and-bound algorithm augmented to handle conditional variables (CondBT-B+B). We performed two separate experiments. For the first experiment, the domain size was fixed at 3, the maximum depth of activity constraints was fixed at 4, and the number of variables was varied from 6 to 20. For the second test, the domain size was fixed at 3, and the depth of activity constraints, *n*, was varied from 1 to 6.



Test 1.) Fixed depth of activity constraints and an increasing # of variables.



Test 2.) Fixed domain size and an increasing depth of activity constraints.

References

[1] Miguel, I. 2001. Dynamic Flexible Constraint Satisfaction and Its Application to AI Planning. PhD diss., The Univ. of Edinburgh.

[2] Keppens, J., 2002. Compositional Ecological Modelling via Dynamic Constraint Satisfaction with Order-of-Magnitude Preferences. Ph.D. Thesis. Univ. of Edinburgh.

[3] Ginsberg, M. 1993. Dynamic Backtracking. *Journal of Artificial Intelligence Research* 1:25--46.

[4] Effinger, R. 2006. Optimal Temporal Planning at Reactive Time Scales via Dynamic Backtracking Branch-and-Bound. S..M.Thesis., MIT.

[5] Gelle, E. and Faltings, B. 2003. Solving mixed and conditional constraint satisfaction problems. *Constraints*, 8(2):107–141.

[6] Sabin, M. 2003. Towards More Efficient Solution of Conditional CSPs. Ph.D. diss. The Univ. of New Hampshire.

***applyNextAC( )*, Conditional Variable Instantiation Function.**

This function simply scans I from beginning to end and returns the first variable, v, which satisfies two conditions:

1.) The variable *must not* belong to the current partial solution, P. (Definition 1)

2.) The variable *must* be on the active variable list, I_A.