A Parameterized Local Consistency for Redundant Modeling in Weighted CSPs*

Y.C. Law, J.H.M. Lee, and M.H.C. Woo

Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{yclaw,jlee,hcwoo}@cse.cuhk.edu.hk

Abstract. The *weighted constraint satisfaction problem* (WCSP) framework is a soft constraint framework which can model many real life optimization or over-constrained problems. While there are many local consistency notions available to speed up WCSP solving, in this paper, we investigate how to effectively combine and channel mutually redundant WCSP models to increase constraint propagation. This successful technique for reducing search space in classical constraint satisfaction has been shown non-trivial when adapted for the WCSP framework. We propose a parameterized local consistency LB(m,Φ), which can be instantiated with *any* local consistency Φ for single models and applied to a combined model with m sub-models, and also provide a simple algorithm to enforce it. We instantiate LB(2,Φ) with different state-of-the-art local consistencies AC*, FDAC*, and EDAC*, and demonstrate empirically the efficiency of the algorithm using different benchmark problems.

1 Introduction

The *weighted constraint satisfaction problem* (WCSP) framework is a well known soft constraint framework for modeling optimization or over-constrained problems. WCSPs are usually solved using backtracking branch and bound search incorporated with constraint propagation that helps reduce the search space. A crucial factor in the solving efficiency is therefore the level of constraint propagation during search. While many state-of-the-art local consistency notions, like AC* [1, 2], FDAC* [3], and EDAC* [4], and their associated algorithms can effectively increase constraint propagation to speed up the search, another common approach is to use *redundant modeling* [5]. The technique, which has been applied successfully in classical constraint satisfaction, is to combine two different models of a problem using *channeling constraints*. Law, Lee, and Woo [6] showed that adapting redundant modeling to WCSPs is a non-trivial task; the node and arc consistency notions have to be refined so as to work on combined models. In this paper, instead of refining a particular local consistency, we propose a *parameterized* local consistency LB(m,Φ) and its associated enforcement algorithm. The advantages of our proposal are three-fold. First, the LB(m,Φ) consistency can be

* We thank the anonymous referees for their constructive comments. The work described in this paper was substantially supported by grants (CUHK4358/02E, CUHK4219/04E, and CUHK4132/07E) from the Research Grants Council of Hong Kong SAR.

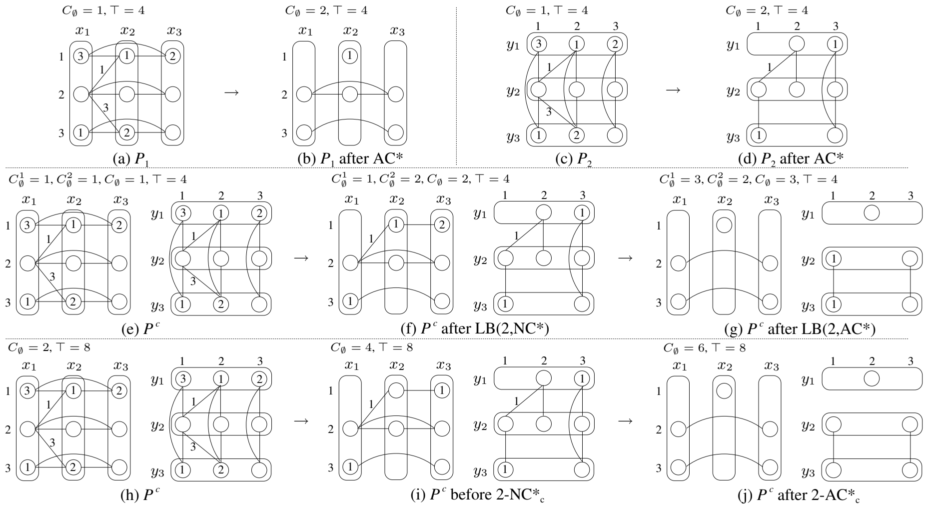

Fig. 1. Enforcing node and arc consistencies on WCSPs $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}^c$

instantiated with *any* local consistency Φ for single models and can be applied to a combined model with m sub-models. Second, the local consistency Φ used for instantiation needs not be refined. Third, enforcing LB(m,Φ) on a combined WCSP model $\mathcal{P}$ achieves *stronger* constraint propagation than enforcing Φ on any individual sub-model of $\mathcal{P}$ alone. We instantiate LB(2,Φ) with AC*, FDAC*, and EDAC* and perform experiments to confirm the feasibility and efficiency of our proposal. Empirical results show that the instantiations always achieve significantly better constraint propagation.

2 Background

WCSPs associate *costs* to tuples [7]. The costs are specified by a *valuation structure* $S(k) = ([0, \ldots, k], \oplus, \geq)$, where k is a natural number, $\oplus$ is defined as $a \oplus b = \min\{k, a+b\}$, and $\geq$ is the standard order among naturals. The minimum and maximum costs are denoted by $\bot = 0$ and $\top = k$ respectively. A binary WCSP is a quadruplet $\mathcal{P} = (k, \mathcal{X}, \mathcal{D}, \mathcal{C})$ with the valuation structure $S(k)$. $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a finite set of *variables* and $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ is a set of finite *domains* for each $x_i \in \mathcal{X}$. An *assignment* $x \mapsto a$ in $\mathcal{P}$ is a mapping from variable x to value $a \in D_x$. A *tuple* is a set of assignments in $\mathcal{P}$. It is *complete* if it contains assignments of all variables in $\mathcal{P}$. $\mathcal{C}$ is a set of unary and binary constraints and a zero-arity constraint. A *unary constraint* involving variable x is a cost function $C_x : D_x \to \{0, \ldots, k\}$. A *binary constraint* involving variables x and y is a cost function $C_{x,y} : D_x \times D_y \to \{0, \ldots, k\}$. A *zero-arity constraint* $C_\emptyset$ is a constant denoting the lower bound of costs in $\mathcal{P}$. Fig. 1(a) shows a WCSP with variables $\{x_1, x_2, x_3\}$ and domains $\{1, 2, 3\}$. We depict the unary costs as labeled nodes and binary costs as labeled edges connecting two assignments. Unlabeled edges have $\top$ cost; $\bot$ costs are not shown for clarity.

The cost of a tuple $\theta = \{x_{i_j} \mapsto a_{i_j} \mid 1 \leq j \leq r\}$ in $\mathcal{P}$ is $\mathcal{V}(\theta) = C_\emptyset \oplus \sum_j C_{x_{i_j}}(a_{i_j}) \oplus \sum_{j<j'} C_{x_{i_j},x_{i_{j'}}}(a_{i_j}, a_{i_{j'}})$. If θ is complete and $\mathcal{V}(\theta) < \top$, θ is a *solution* of $\mathcal{P}$. Solving a WCSP is to find a solution θ with minimized $\mathcal{V}(\theta)$, which is NP-hard. The WCSP in Fig. 1(a) has an optimal solution $\{x_1 \mapsto 2, x_2 \mapsto 1, x_3 \mapsto 3\}$ with cost 3, as $C_\emptyset \oplus C_{x_1}(2) \oplus C_{x_2}(1) \oplus C_{x_3}(3) \oplus C_{x_1,x_2}(2,1) \oplus C_{x_1,x_3}(2,3) \oplus C_{x_2,x_3}(1,3) = 1 \oplus \bot \oplus 1 \oplus \bot \oplus 1 \oplus \bot \oplus \bot = 3$. A WCSP reduces to a classical CSP if each cost in the WCSP is either $\bot$ or $\top$. Two WCSPs are *equivalent* if they have the same variables and for every complete tuple θ, $\mathcal{V}(\theta)$ is the same for both WCSPs.

WCSPs can be solved by backtracking *branch and bound* (B&B) search that maintains some form of local consistency at each search node. A *local consistency* is simply some properties of a WCSP. *Enforcing* a local consistency Φ on a WCSP $\mathcal{P}$ means transforming $\mathcal{P}$ to an equivalent WCSP $\mathcal{P}'$ that satisfies the properties specified by Φ. $\mathcal{P}'$ is usually simplified in the sense that either some domain values of the variables are pruned or the lower bound is increased. In the following, we give the definitions of two common local consistencies NC* [1, 2] and AC* [1, 2] for WCSPs.

Definition 1. *A variable x_i in a WCSP $\mathcal{P}$ is* node consistent *(NC*) if (1) for all values $a \in D_{x_i}$, $C_\emptyset \oplus C_{x_i}(a) < \top$ and (2) there exists a value $a \in D_{x_i}$ such that $C_{x_i}(a) = \bot$. A WCSP $\mathcal{P}$ is NC* if every variable in $\mathcal{P}$ is NC*.*

Definition 2. *Given a constraint C_{x_i,x_j} in a WCSP $\mathcal{P}$, a value $b \in D_{x_j}$ is a* support *for a value $a \in D_{x_i}$ if $C_{x_i,x_j}(a, b) = \bot$. A variable x_i is* arc consistent *(AC*) if every value $a \in D_{x_i}$ has a support in every constraint C_{x_i,x_j}. A WCSP $\mathcal{P}$ is* arc consistent *(AC*) if every variable in $\mathcal{P}$ is AC*.*

NC* can be enforced by sending some costs from the *unary* constraints to $C_\emptyset$ and pruning the node-inconsistent values [1, 2]. Similarly, AC* can be enforced by sending some costs from the *binary* constraints to the unary constraints, and then relying on NC* to further move the costs to $C_\emptyset$ or prune any values [1, 2]. Thus, $C_\emptyset$ can be seen as a lower bound of costs of a WCSP. For example, the WCSP in Fig. 1(b) is AC* and equivalent to the one in Fig. 1(a). It has some domain values pruned and the lower bound $C_\emptyset$ is increased. In backtracking B&B search, whenever $C_\emptyset$ is increased to $\top$, we cannot continue to extend a tuple to obtain a solution, and hence a backtrack (or sometimes called a fail) is triggered. Also, whenever a solution θ is found, the $\top$ value will be set to $\mathcal{V}(\theta)$ to continue the search, ensuring that the next solution found must have a better cost than θ. At the end of the search, the last found solution is optimal.

3 A Parameterized Local Consistency for Redundant Modeling

In this section, we first describe how we can obtain redundant WCSP models. Then, we suggest a method to combine mutually redundant models into one model. A parameterized local consistency LB(m,Φ), which maintains the same local consistency Φ on each sub-model, and its enforcement algorithm are proposed for the suggested combined model. A bijective channel function, implemented using table lookup, is used to map assignments in one sub-model to the other. Theoretical comparisons are made between the proposed consistency and an existing approach for redundant modeling.

3.1 Obtaining a Redundant Model

Deriving multiple classical CSP models for the same problem is common, although not trivial. It is even more difficult to obtain an alternative model in WCSP since each problem solution is associated with a cost and we have to ensure the same cost distribution on the solutions of the redundant WCSP models. Two WCSPs $\mathcal{P}_1$ and $\mathcal{P}_2$ are *mutually redundant* [6] if (1) there is a bijection g between the two sets of all solutions of $\mathcal{P}_1$ and $\mathcal{P}_2$ and (2) for every solution θ of $\mathcal{P}_1$, the associated costs of solution θ of $\mathcal{P}_1$ and solution $g(\theta)$ of $\mathcal{P}_2$ are the same, i.e., $\mathcal{V}(\theta) = \mathcal{V}(g(\theta))$.

Based on these two requirements, Law et al. [6] proposed *generalized model induction* that generates mutually redundant *permutation WCSPs* from a given one. In a permutation WCSP, the variables in a solution must take all-different values. Given a WCSP $\mathcal{P} = (k, \mathcal{X}, \mathcal{D}_\mathcal{X}, \mathcal{C}_\mathcal{X})$. A *channel function* maps assignments in $\mathcal{P}$ to those in another set of variables. If $\mathcal{P}$ is a permutation WCSP, without loss of generality, we always have the bijective channel function $f(x_i \mapsto j) = y_j \mapsto i$. The constraints $\mathcal{C}_\mathcal{Y}$ in the *induced model* $\mathcal{P}' = (k, \mathcal{Y}, \mathcal{D}_\mathcal{Y}, \mathcal{C}_\mathcal{Y})$ are defined such that for $1 \le a, i \le n$, $C_{y_a}(i) = C_{x_i}(a)$, and for $1 \le a, b, i, j \le n$, $C_{y_a,y_b}(i,j) = C_{x_i,x_j}(a,b)$ if $i \ne j$; and $C_{y_a,y_b}(i,j) = \top$ otherwise. Note that the induced model $\mathcal{P}'$ must be a permutation WCSP, since $C_{y_a,y_b}(i,i) = \top$ for all $1 \le a, b, i \le n$. Fig. 1(c) shows the induced model $\mathcal{P}_2$ of $\mathcal{P}_1$ in Fig. 1(a). In the example, we have, say, the unary cost $C_{y_1}(2) = C_{x_2}(1) = 1$ and the binary cost $C_{y_2,y_3}(1,2) = C_{x_1,x_2}(2,3) = 3$.

3.2 Combining Mutually Redundant Models

Given a problem, we can always formulate it into different mutually redundant WCSP models. Redundant modeling is a common technique to take advantages of each model by combining them into one model using *channeling constraints*, which are constraints that set forth the relationship between the variables in any two models.

Consider m mutually redundant models $\mathcal{P}_i = (k, \mathcal{X}_i, \mathcal{D}_i, \mathcal{C}_i)$ for $1 \le i \le m$ of the same problem, we propose to connect them using a set of channeling constraints $\mathcal{C}^c$ to give a *combined model* $\mathcal{P}^c = (k, \bigcup_i \mathcal{X}_i, \bigcup_i \mathcal{D}_i, \bigcup_i \mathcal{C}_i \cup \mathcal{C}^c)$. In addition to a single $C_\emptyset$ for a combined model $\mathcal{P}^c$, we now associate each sub-model $\mathcal{P}_i$ with a *local zero-arity constraint* $C^i_\emptyset$ to denote the local lower bound of costs in $\mathcal{P}_i$. Since each sub-model $\mathcal{P}_i$ has its own $C^i_\emptyset$, enforcing NC* sends the costs of the unary constraints in sub-model $\mathcal{P}_i$ to $C^i_\emptyset$ of $\mathcal{P}^c$. Besides, $\mathcal{C}^c$ are hard constraints connecting sub-models in $\mathcal{P}^c$. For example, a channeling constraint is typically of the form $x_i = p \Leftrightarrow y_j = q$. It has the cost function $C_{x_i,y_j}(a,b) = \perp$ if $a = p \Leftrightarrow b = q$; and $C_{x_i,y_j}(a,b) = \top$ otherwise. In addition, the value k remains unchanged for each sub-model $\mathcal{P}_i$ in $\mathcal{P}^c$. The lower bound $C_\emptyset$ of $\mathcal{P}^c$ takes the maximum value of all local lower bounds $C^i_\emptyset$ of $\mathcal{P}_i$ (i.e., $C_\emptyset = \max_i\{C^i_\emptyset\}$). Based on the combined model, we denote a complete tuple in a sub-model $\mathcal{P}_i$ as a *semi-complete tuple* θ_i. Thus, similar to $C_\emptyset$, the cost of a complete tuple θ in $\mathcal{P}^c$ takes the maximum value of all semi-complete tuples θ_i in $\mathcal{P}_i$ (i.e., $\mathcal{V}(\theta) = \max_i\{\mathcal{V}(\theta_i)\}$). Fig. 1(e) shows the combined model $\mathcal{P}^c$ of $\mathcal{P}_1$ and $\mathcal{P}_2$ in Figs. 1(a) and 1(c) respectively.

function LB$(m, \Phi, \mathcal{P}^c)$

1\. **repeat**

2\.   **for** each sub-model $\mathcal{P}_s$ of $\mathcal{P}^c$ **do** enforce Φ on $\mathcal{P}_s$;

3\.   **for** each pair of sub-models $\mathcal{P}_s, \mathcal{P}_t$ of $\mathcal{P}^c$ $(s \neq t)$ **do**

4\.   **for** each $x_{s,i} \in \mathcal{X}_s$ **do**

5\.   **for** each $a \in D_{x_{s,i}}$ **do**

6\.   **if** $b \notin D_{x_{t,j}}$ where $x_{t,j} \mapsto b = f_{s,t}(x_{s,i} \mapsto a)$ **then** remove a from $D_{x_{s,i}}$;

7\. **until** $\mathcal{P}^c$ remains unchanged;

Fig. 2. Algorithms for enforcing LB(m,Φ)

3.3 Enforcing Consistency on Combined Models

In redundant modeling, mutually redundant models are connected using a set $\mathcal{C}^c$ of channeling constraints, which are mainly used to transmit pruning and variable instantiation information between sub-models. If we rely on the generic AC* or other local consistency algorithms to propagate the channeling constraints, it will cause a large overhead on the execution. Consider m mutually redundant WCSP sub-models in $\mathcal{P}^c$. With a set of channeling constraints of the form $x = a \Leftrightarrow y = b$ connecting sub-models $\mathcal{P}_i$ and $\mathcal{P}_j$, we can define a bijective channel function that maps assignments in $\mathcal{P}_i$ to those in $\mathcal{P}_j$, giving, for example, $f(x \mapsto a) = y \mapsto b$. A channel function can be implemented using table lookup in $O(1)$ time and can be used to aid the propagation of the channeling constraints more efficiently. During constraint propagation, when a value $a \in D_x$ is pruned from sub-model $\mathcal{P}_i$, we can use the channel function to immediately know that the corresponding value $b \in D_y$ should be pruned as well. Similarly, when a variable x in $\mathcal{P}_i$ is bound to the value a, according to the channel function, we can also know that the corresponding variable y in $\mathcal{P}_j$ should be bound to the value b.

Given a combined model $\mathcal{P}^c$ with m mutually redundant sub-models $\mathcal{P}_i$. We can enforce local consistency Φ on each sub-model $\mathcal{P}_i$ of $\mathcal{P}^c$ and use the channel function to transmit instantiation and pruning information between $\mathcal{P}_i$ to ensure that the bijective mapping between assignments of any two sub-models $\mathcal{P}_i$ and $\mathcal{P}_j$ for $1 \leq i < j \leq m$ is maintained. Based on these ideas, we proposed a parameterized local consistency LB(m,Φ) for combined models $\mathcal{P}^c$ with m mutually redundant sub-models $\mathcal{P}_i$. Note that Φ can be any local consistency that can be applied to a single WCSP model.

Definition 3. *Let $\mathcal{P}^c$ be a combined model of m mutually redundant sub-models $\mathcal{P}_s$ for $1 \leq s \leq m$, Φ be a local consistency, and $f_{s,t}$ be a bijective channel function from assignments of $\mathcal{P}_s$ to assignments of $\mathcal{P}_t$ for all $1 \leq s < t \leq m$. $\mathcal{P}^c$ is said to be* LB(m,Φ) *if: (1) all sub-models $\mathcal{P}_s$ are Φ, and (2) for all assignments $x_{s,i} \mapsto a$ of $\mathcal{P}_s$, $a \in D_{x_{s,i}} \Leftrightarrow b \in D_{x_{t,j}}$, where $f_{s,t}(x_{s,i} \mapsto a) = x_{t,j} \mapsto b$.*

Consider the combined model $\mathcal{P}^c$ in Fig. 1(e). It is not LB(2,AC*) since both sub-models $\mathcal{P}_1$ and $\mathcal{P}_2$ are not AC*. After enforcing AC* on each sub-model and sharing the pruning information between sub-models, Fig. 1(g) gives an equivalent combined model $\mathcal{P}^{c\prime}$, which is now LB(2,AC*).

LB(m,Φ) can be enforced using a simple algorithm shown in Fig. 2. In this algorithm, we enforce Φ on each sub-model (line 2). This ensures that all sub-models $\mathcal{P}_s$ satisfy the Φ property (condition 1). For condition 2, if there is a value $a \in D_{x_{s,i}}$ being pruned in one sub-model $\mathcal{P}_s$, the corresponding value $b \in D_{x_{t,j}}$ obtained via the channel function will also be pruned in other sub-models $\mathcal{P}_t$ for $1 \leq t \leq m$ and $s \neq t$ (lines 3–6). The algorithm repeats until there are no more changes in any sub-models, and $\mathcal{P}^c$ is then made LB(m,Φ). Since each sub-model $\mathcal{P}_s$ has its local lower bound $C^s_\emptyset$, unary constraints are projected towards its own $C^s_\emptyset$ when enforcing NC*. For example, unary constraints in $\mathcal{P}_1$ are projected to $C^1_\emptyset$ and those in $\mathcal{P}_2$ are projected to $C^2_\emptyset$. During constraint propagation, when the global lower bound $C_\emptyset$ of the combined model $\mathcal{P}^c$ reaches the global upper bound $\top$ (i.e., $C_\emptyset = \max_s\{C^s_\emptyset\} = \top$), this means that there exists at least one sub-model $\mathcal{P}_s$ in which its local lower bound $C^s_\emptyset$ is increased to $\top$, and we cannot extend a tuple of this sub-model to obtain a solution. Since all the sub-models are mutually redundant to each other, the other sub-models will also lead to failure. Therefore, a backtrack is triggered in the search. The following theorem states that the algorithm in Fig. 2 enforces LB(m,Φ).

Theorem 1. *Let $\mathcal{P}^c$ be a combined model of m WCSP sub-models $\mathcal{P}_s$ for $1 \leq s \leq m$, and Φ be any local consistency. The LB$(m, \Phi, \mathcal{P}^c)$ algorithm transforms $\mathcal{P}^c$ into an equivalent combined model $\mathcal{P}^{c\prime}$.*

Proof. When a combined model $\mathcal{P}^c$ is passed to the LB$(m, \Phi, \mathcal{P}^c)$ algorithm, enforcing Φ on each sub-model $\mathcal{P}_i$ transforms $\mathcal{P}_i$ to an equivalent sub-model $\mathcal{P}'_i$. Furthermore, the mutual redundancy of two sub-models $\mathcal{P}_s$ and $\mathcal{P}_t$ guarantees that if value b is not in $D_{x_{t,j}}$, then value a must not be in the domain of $x_{s,i}$, where $x_{t,j} \mapsto b = f_{s,t}(x_{s,i} \mapsto a)$. Thus, removing value a from $D_{x_{s,i}}$ in line 6 will not remove any values that belong to a solution of the combined model $\mathcal{P}^c$. Hence, upon termination of the algorithm, the transformed model $\mathcal{P}^{c\prime}$ is equivalent to the input model $\mathcal{P}^c$. □

Following Debruyne and Bessiere [8], we define some notions to compare the propagation strength of two local consistencies Φ_1 and Φ_2. Φ_1 is said to be *stronger than* [8] Φ_2 if in any WCSP in which Φ_1 holds, then Φ_2 holds. Φ_1 is said to be *strictly stronger than* [8] Φ_2 if (1) Φ_1 is stronger than Φ_2 and (2) there exists a WCSP where Φ_2 holds but Φ_1 does not hold.

Theorem 2. *Let $\mathcal{P}^c$ be a combined model of m sub-models $\mathcal{P}_s$ for $1 \leq s \leq m$. Enforcing LB(m,Φ) on $\mathcal{P}^c$ is strictly stronger than enforcing Φ on any $\mathcal{P}_s$.*

Proof. By definition 3, $\mathcal{P}^c$ is LB(m,Φ) if all sub-models $\mathcal{P}_s$ for $1 \leq s \leq m$ are Φ. This shows that LB(m,Φ) is stronger than Φ. To show strictness, consider the model in Fig. 1(e) which is a combined model of $\mathcal{P}_1$ and $\mathcal{P}_2$ in Figs. 1(a) and 1(c) respectively. Enforcing AC* on $\mathcal{P}_1$ and $\mathcal{P}_2$ individually yields the models in Figs. 1(b) and 1(d) respectively. However, the combined model of these two sub-models is not LB(2,AC*). In fact, enforcing LB(2,AC*) results in the model in Fig. 1(g). Hence the result. □

Note that unlike classical CSPs, enforcing AC* on a WCSP can result in more than one possible outcome, depending on the order of the domain values to be pruned and the constraints to be handled in an algorithm [2]. Therefore, although we have the "strictly stronger" notion, when comparing a LB(m,AC*) combined model and a AC* sub-model, we cannot guarantee that the domain of a variable in the combined model must be a subset of that in the single sub-model. Nonetheless, such theoretical comparison is still worthwhile as it shows that enforcing one local consistency can generally prune more domain values than enforcing another.

3.4 Comparison with Existing Work

Given m mutually redundant WCSPs $\mathcal{P}_s = (k_s, \mathcal{X}_s, \mathcal{D}_{\mathcal{X}s}, \mathcal{C}_s)$ for $1 \leq s \leq m$, Law et al. [6] suggested another way to form a combined model. Instead of taking the same k and $\top$ as each sub-model has, the combined model $\mathcal{P}^c = (\sum_s k_s, \bigcup_s \mathcal{X}_s, \bigcup_s \mathcal{D}_s, \bigcup_s \mathcal{C}_s \cup \mathcal{C}^c)$ uses the values k and $\top$ which are the sum of all values k_s and $\top_s$ respectively of its sub-models $\mathcal{P}_s$. For example, Fig. 1(h) gives a combined model $\mathcal{P}^c$ of two mutually redundant models $\mathcal{P}_1$ and $\mathcal{P}_2$ in Figs. 1(a) and 1(c) respectively, with $C_\emptyset = 1 \oplus 1 = 2$ and $\top = 4 \oplus 4 = 8$. Besides, $\mathcal{P}^c$ has only one lower bound $C_\emptyset$; there are no individual lower bound for sub-models. Any cost that is sent from the unary constraints in any sub-model all goes to $C_\emptyset$. Since $C_\emptyset$ and $\top$ are shared among sub-models in $\mathcal{P}^c$, the local consistency has to be refined for combined models. Law et al. [6] proposed new notions of node consistency m-NC$^*_\text{c}$ and arc consistency m-AC$^*_\text{c}$ to transmit pruning and cost movement information between sub-models. In our proposed combined model, sub-models have their own local lower bound; pruning information are transmitted via the channel function. Thus, refinements of local consistencies are not required.

When enforcing m-NC$^*_\text{c}$ and m-AC$^*_\text{c}$, not only the instantiation and pruning information but also the cost projection information is transmitted between sub-models in the combined model. Transmitting cost projection information can further discover and remove more node inconsistent values or increase the global lower bound. Thus, enforcing 2-NC$^*_\text{c}$ and 2-AC$^*_\text{c}$ achieves more constraint propagation than enforcing LB(2,NC*) and LB(2,AC*) respectively.

Theorem 3. *Let $\mathcal{P}^c$ be a combined model of m mutually redundant sub-models $\mathcal{P}_s$ for $1 \leq s \leq m$. Enforcing m-NC$^*_\text{c}$* (resp. *m-AC$^*_\text{c}$) on $\mathcal{P}^c$ is* strictly stronger *than enforcing LB(m,NC*)* (resp. *LB(m,AC*)) on $\mathcal{P}^c$.*

Due to space limitation, we do not provide the formal definitions of m-NC$^*_\text{c}$ and m-AC$^*_\text{c}$ and thus the proof. Instead, we give an example to demonstrate the theorem. Consider the problem in Fig. 1. Figs. 1(f) and 1(i) give two combined WCSPs after enforcing LB(2,NC*) and 2-NC$^*_\text{c}$ respectively. We can observe that enforcing 2-NC$^*_\text{c}$ achieves more domain prunings and a greater lower bound (i.e., $C_\emptyset = 4 > C^1_\emptyset \oplus C^2_\emptyset = 1 \oplus 2 = 3$). Similarly, the WCSP in Fig. 1(j) is 2-AC$^*_\text{c}$ and has a greater lower bound than the LB(2,AC*) WCSP in Fig. 1(g).

4 Experiments

To evaluate the effectiveness and efficiency of the combined models, we implement our approach in ToolBar[1], a branch and bound WCSP solver maintaining local consistencies at each search tree node. Three benchmarks, n-queens problem, knight's tour problem, and Langford's problem are modeled as WCSPs to test our approach. Comparisons are made among AC* [1, 2], FDAC* [3], and EDAC* [4] on a single model $\mathcal{P}$, 2-AC$^*_\text{c}$ [6] on a combined model $\mathcal{P}^c$, and LB(2,AC*), LB(2,FDAC*), and LB(2,EDAC*) on a combined model $\mathcal{P}^c$ proposed in this paper. All combined models $\mathcal{P}^c$ contain a single

[1] Available at http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro.

model $\mathcal{P}$ and its induced model $\mathcal{P}'$, generated automatically using generalized model induction [6], as sub-models.

The experiments are run on a Sun Blade 2500 (2 $\times$ 1.6GHz US-IIIi) workstation with 2GB memory. We use the *dom/deg* variable ordering heuristic which chooses the variable with the smallest ratio of domain size to future degree. Values are chosen using the *smallest-cost-first* and the *dual-smallest-domain-first* [5] heuristics on $\mathcal{P}$ and $\mathcal{P}^c$ respectively. The initial $\top$ value provided to the solver is n^2, where n is the number of variables in a model. We generate ten random models for each soft instance and report the average number of fails (i.e., the number of backtracks occurred in solving a model) and CPU time in seconds to find the first optimal solution. In the tables, the first column shows the problem instances; those marked with "*" have a $\perp$ optimal cost. The subsequent columns show the results of enforcing various local consistencies on either $\mathcal{P}$ or $\mathcal{P}^c$. The best number of fails and CPU time among the results for each instance are highlighted in bold. A cell labeled with "-" denotes a timeout after two hours.

Table 1(a) shows the experimental results of soft n-queens problem. The n-queens problem is to place n queens on a $n \times n$ chessboard so that no two queens are placed on the same row, same column, or same diagonal. To model the problem into a WCSP $\mathcal{P}_1 = (\mathcal{X}, \mathcal{D}_\mathcal{X}, \mathcal{C}_\mathcal{X})$, we use n variables $\mathcal{X} = \{x_1, \ldots, x_n\}$. Each variable x_i denotes the row position of queen i in column i of the chessboard. The domains of the variables are thus $\{1, \ldots, n\}$. The constraints can be expressed using these variables accordingly.

Since the n-queens problem has many solutions, we assert preferences among the solutions by assigning to each allowed binary tuple a random cost from $\perp$ to $\top$ inclusive for soft instances. From the results, we observe that solving a combined model $\mathcal{P}^c$ achieves fewer number of fails and mostly faster runtime than solving a single model $\mathcal{P}$, which means that 2-AC$^*_\mathrm{c}$, LB(2,AC*), LB(2,FDAC*), and LB(2,EDAC*) do more prunings than AC*, FDAC*, and EDAC* and reduce more search space. LB(2,AC*) is the most efficient for the larger instances. Although 2-AC$^*_\mathrm{c}$ is stronger than LB(2,AC*), it gives a longer runtime than LB(2,AC*) even it achieves smaller number of fails in solving the soft instances. This is mainly due to the overhead of cost projection transmitting between sub-models in 2-AC$^*_\mathrm{c}$. LB(2,EDAC*) always has the fewest number of fails since EDAC* is stronger than AC* and FDAC*. However, the runtime of LB(2,EDAC*) is not the fastest, as the extra prunings cannot compensate for the overhead of EDAC* over AC*. According to Larrosa (personal communication), the implementation of EDAC* is optimized for small variable domains. The same applies to the timing of AC* and EDAC* on $\mathcal{P}$. We perform experiments on the classical n-queens problem as well, and the results are similar to those of the soft cases. Due to space limitation, we omit the details.

Tables 1(b) and 1(c) show the results of classical and soft knight's tour problem. This problem is to find a sequence of moves by a knight on a $m \times n$ chessboard so that each square on the chessboard is traversed exactly once. We model this problem into a WCSP $\mathcal{P}$ using mn variables $\mathcal{X} = \{x_1, \ldots, x_{mn}\}$. Each variable x_i denotes a square on the chessboard and the domains of the variables $\{1, \ldots, mn\}$ denote the order of a move sequence. There are constraints to ensure that the knight takes a valid move (i.e., moves either one square horizontally and two squares vertically, or two squares horizontally and one square vertically). Similar to the n-queens problem, we soften the

Table 1. Experimental results on solving n-queens and knight's tour problems

(a) Soft n-queens problem

n	AC* fail	AC* time	FDAC* fail	FDAC* time	EDAC* fail	EDAC* time	2-AC$_c^*$ fail	2-AC$_c^*$ time	LB(2,AC*) fail	LB(2,AC*) time	LB(2,FDAC*) fail	LB(2,FDAC*) time	LB(2,EDAC*) fail	LB(2,EDAC*) time
15*	5608	**0.46**	5044	0.50	4959	0.63	2759	0.58	2803	0.58	2546	0.65	**2488**	0.77
16*	10611	**0.96**	9566	1.03	9382	1.29	5340	1.23	5427	1.21	4873	1.37	**4769**	1.61
17*	17369	**1.65**	15139	1.80	14703	2.25	7291	1.85	7429	1.81	6668	2.08	**6493**	2.46
18*	58375	**6.23**	51073	6.76	49309	8.29	26370	7.73	26762	7.48	24479	8.49	**23880**	9.95
19*	81022	**9.20**	70179	10.05	67341	12.41	32921	9.91	33681	9.60	29197	11.07	**27919**	12.81
20*	172220	**21.65**	150939	23.50	145062	28.94	94256	32.44	95941	31.16	86345	35.83	**83442**	42.09
21*	535145	73.28	463225	79.78	441403	97.35	178540	65.39	182731	**62.39**	159087	72.03	**151279**	83.29
22*	1287717	196.07	1130132	211.09	1078297	257.76	470117	200.77	478708	**188.64**	432133	214.43	**415874**	249.10
23*	4780028	810.60	4256142	868.88	4076254	1060.10	1274332	617.77	1294032	**575.07**	1183741	650.64	**1143417**	755.17
24*	11079154	2042.66	9928478	2182.87	9518203	2637.39	3256536	1738.86	3307770	**1602.52**	3029570	1806.44	**2921698**	2081.07

(b) Classical knight's tour problem

(m,n)	AC* fail	AC* time	FDAC* fail	FDAC* time	EDAC* fail	EDAC* time	2-AC$_c^*$ fail	2-AC$_c^*$ time	LB(2,AC*) fail	LB(2,AC*) time	LB(2,FDAC*) fail	LB(2,FDAC*) time	LB(2,EDAC*) fail	LB(2,EDAC*) time
(3,5)	672	0.07	672	0.07	672	0.10	**210**	0.08	**210**	0.08	**210**	**0.06**	**210**	0.08
(3,6)	4416	0.45	4416	0.53	4416	0.70	**816**	0.36	**816**	0.34	**816**	**0.29**	**816**	0.33
(3,7)*	545	**0.06**	545	0.07	545	0.10	**158**	0.08	**158**	0.08	**158**	0.08	**158**	0.08
(3,8)*	2657	0.33	2657	0.38	2657	0.50	**732**	0.35	**732**	0.33	**732**	**0.29**	**732**	0.33
(3,9)*	11535	1.66	11535	1.94	11535	2.61	**2005**	1.29	**2005**	1.21	**2005**	**1.07**	**2005**	1.20
(3,10)*	72183	11.79	72183	13.74	72183	18.72	**10628**	8.25	**10628**	7.71	**10628**	**6.72**	**10628**	7.49
(3,11)*	13225	2.17	13225	2.46	13225	3.39	**1878**	1.79	**1878**	1.60	**1878**	**1.31**	**1878**	1.48
(3,12)*	2349445	467.80	2349445	550.14	2349445	743.09	**212657**	269.15	**212657**	240.85	**212657**	**201.84**	**212657**	228.67
(3,13)	766731	172.10	766731	202.52	766731	274.80	**49684**	94.08	**49684**	83.70	**49684**	**68.33**	**49684**	74.73
(3,14)	-	-	-	-	-	-	-	-	**3679309**	6816.84	**3679309**	**5550.03**	**3679309**	6249.42
(3,15)	1498214	367.45	1498214	426.99	1498214	588.97	**58889**	147.85	**58889**	134.74	**58889**	**107.39**	**58889**	118.52
(4,5)*	336	**0.03**	336	**0.03**	336	0.04	**77**	0.04	**77**	**0.03**	**77**	0.04	**77**	0.04
(4,6)*	7914	0.68	7914	0.76	7914	1.04	**1415**	0.56	**1415**	0.53	**1415**	**0.46**	**1415**	0.51
(4,7)*	40344	4.19	40344	4.62	40344	6.55	**5526**	2.70	**5526**	2.51	**5526**	**2.07**	**5526**	2.30
(4,8)*	1183823	145.18	1183823	161.48	1183823	228.35	**76661**	53.58	**76661**	49.00	**76661**	**39.74**	**76661**	43.63
(4,9)	462506	67.28	462506	74.85	462506	105.82	**33575**	31.86	**33575**	28.07	**33575**	**23.35**	**33575**	25.66
(4,10)	29317520	5166.23	29317520	5769.60	-	-	**971162**	1469.12	**971162**	1316.55	**971162**	**1034.83**	**971162**	1144.33

(c) Soft knight's tour problem

(m,n)	AC* fail	AC* time	FDAC* fail	FDAC* time	EDAC* fail	EDAC* time	2-AC$_c^*$ fail	2-AC$_c^*$ time	LB(2,AC*) fail	LB(2,AC*) time	LB(2,FDAC*) fail	LB(2,FDAC*) time	LB(2,EDAC*) fail	LB(2,EDAC*) time
(3,5)	685	**0.08**	672	0.11	683	0.19	**210**	0.10	**210**	0.09	**210**	0.09	**210**	0.12
(3,6)	4482	0.52	4418	0.81	4463	1.46	**816**	0.39	**816**	**0.37**	**816**	0.43	**816**	0.62
(3,7)	21005	2.94	13629	3.85	13785	5.74	4576	2.13	4750	**2.09**	3571	2.42	**3536**	3.49
(3,8)	112979	19.93	61182	24.30	60967	34.70	21452	**11.46**	22971	11.48	14320	12.57	**14022**	17.96
(3,9)	679347	140.38	341322	166.53	339664	238.65	113032	72.89	121562	**72.20**	73220	79.67	**71996**	118.24
(3,10)	3822955	952.94	1720858	1061.83	1697349	1505.17	511042	431.53	557168	**423.39**	299257	440.19	**294434**	651.79
(3,11)	-	-	-	-	-	-	2319839	2587.78	2590939	2579.96	1236203	**2432.43**	**1216082**	3579.21
(4,5)	45516	4.86	27468	5.85	27374	8.72	5515	1.96	5719	**1.94**	4165	2.14	**4111**	3.05
(4,6)	863270	116.09	415083	129.63	407198	187.90	98151	**42.53**	104256	42.71	61805	43.33	**60522**	63.25
(4,7)	12899196	2335.48	5005880	2339.55	4831495	3248.17	1344421	815.54	1468226	814.30	719927	**742.79**	**702569**	1085.02

problem by assigning a random cost to each allowed binary tuple for soft instances. The random cost is assigned from $\perp$ to mn inclusively. In this problem, the induced model $\mathcal{P}'$ is a better model than $\mathcal{P}$. Therefore, we use $\mathcal{P}'$ as a basic model instead of $\mathcal{P}$.

For both classical and soft instances, enforcing local consistencies on $\mathcal{P}^c$, similar to the n-queens problem, achieves smaller number of fails and shorter runtime than those on $\mathcal{P}$. For the classical instance $(3, 14)$, using AC*, FDAC*, EDAC*, and even 2-AC$_c^*$ cannot solve the problem before timeout, but using LB(2,AC*), LB(2,FDAC*), and LB(2,EDAC*) can. Among the local consistencies on $\mathcal{P}^c$, LB(2,FDAC*) always achieves the fastest runtime. LB(2,AC*), LB(2,FDAC*), LB(2,EDAC*), and 2-AC$_c^*$ on $\mathcal{P}^c$ all result in the same number of fails because in the classical cases, EDAC* and FDAC* degenerate to AC*. In soft cases, LB(2,AC*) and 2-AC$_c^*$ have similar runtime but 2-AC$_c^*$ has a slightly smaller number of fails. LB(2,EDAC*) achieves the smallest number of fails in most instances, while LB(2,FDAC*) has the shortest runtime in large and difficult instances.

For the Langford's problem[2] (which we do not present the experimental data), using LB(2,Φ) is not always faster than using 2-AC_c^*, although both are much better than using single models. In classical cases, LB(2,AC*) and LB(2,FDAC*) have a slightly faster runtime than 2-AC_c^* for the large instances, but they are slightly slower than 2-AC_c^* in the soft cases. Although LB(2,Φ) does not always perform better than 2-AC_c^*, it has the flexibility of choosing *any* local consistency for single models to solve a problem.

5 Conclusion

Combining mutually redundant WCSP models is a non-trivial task. In this paper, we have proposed a parameterized local consistency LB(m,Φ) that is specifically designed for combined models to increase constraint propagation. One important advantage of our proposal is that LB(m,Φ) can be used with *any* number of sub-models and *any* existing or future local consistencies that are targeted for solving single models, making our proposal highly flexible. Experimental results confirm that LB(2,Φ) performs well when instantiated with the state-of-the-art AC*, FDAC*, and EDAC*. The search space is significantly reduced when compared with using single models. LB(2,Φ) is also competitive with, if not better than, 2-AC_c^* in the benchmarks.

Redundant modeling for WCSPs is a new concept which has much scope for future work. In classical CSPs, it is common to have one model in integer variables while another in set variables. We can investigate if WCSPs with different types of variables can be combined and solved. In this paper, we focus on permutation problems. We are interested in knowing how generalized model induction can generate induced models of non-permutation WCSPs and if the concept of redundant modeling can be extended to combined models connected with non-bijective channeling constraints.

References

1. Larrosa, J.: Node and arc consistency in weighted CSP. In: Proc. of AAAI'02. (2002) 48–53
2. Larrosa, J., Schiex, T.: Solving weighted CSP by maintaining arc consistency. Artificial Intelligence **159**(1-2) (2004) 1–26
3. Larrosa, J., Schiex, T.: In the quest of the best form of local consistency for weighted CSP. In: Proc. of IJCAI'03. (2003) 239–244
4. de Givry, S., Heras, F., Zytnicki, M., Larrosa, J.: Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In: Proc. of IJCAI'05. (2005) 84–89
5. Cheng, B.M.W., Choi, K.M.F., Lee, J.H.M., Wu, J.C.K.: Increasing constraint propagation by redundant modeling: an experience report. Constraints **4**(2) (1999) 167–192
6. Law, Y.C., Lee, J.H.M., Woo, M.H.C.: Speeding up weighted constraint satisfaction using redundant modeling. In: Proc. of AI'06. (2006) 59–68
7. Schiex, T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: hard and easy problems. In: Proc. of IJCAI'95. (1995) 631–637
8. Debruyne, R., Bessière, C.: Some practicable filtering techniques for the constraint satisfaction problem. In: Proc. of IJCAI'97. (1997) 412–417

[2] Prob024 in CSPLib, available at http://www.csplib.org.