Parameterized Heuristics for Incomplete Weighted CSPs with Elicitation Costs

Atena M. Tabakhi
Washington University in St. Louis
St. Louis, MO, USA
amtabakhi@wustl.edu

William Yeoh
Washington University in St. Louis
St. Louis, MO, USA
wyeoh@wustl.edu

Makoto Yokoo
Kyushu University
Fukuoka, Japan
yokoo@inf.kyushu-u.ac.jp

ABSTRACT

Weighted Constraint Satisfaction Problems (WCSPs) are an elegant paradigm for modeling combinatorial optimization problems. A key assumption in this model is that all constraints are specified or known a priori, which does not hold in some applications where constraints may encode preferences of human users. *Incomplete WCSPs* (IWCSPs) extend WCSPs by allowing some constraints to be partially specified, and they can be elicited from human users during the execution of IWCSP algorithms. Unfortunately, existing approaches assume that the elicitation of preferences does not incur any additional cost. This assumption is unrealistic as human users are likely bothered by repeated elicitations and will refuse to provide an unbounded number of preferences. Therefore, we propose the *IWCSP with Elicitation Cost* (IWCSP+EC) model, which extends IWCSPs to include *elicitation costs*, as well as three parameterized heuristics that allow users to trade off solution quality for fewer elicited preferences and faster computation times. They provide theoretical quality guarantees for problems where elicitations are free. Our model and heuristics thus extend the state of the art in constraint reasoning to better model and solve agent-based applications with user preferences.

KEYWORDS

Weighted CSPs; Incomplete Weighted CSPs; Preference Elicitation

ACM Reference Format:
Atena M. Tabakhi, William Yeoh, and Makoto Yokoo. 2019. Parameterized Heuristics for Incomplete Weighted CSPs with Elicitation Costs. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

The importance of constraint reasoning in agent-based systems is outlined by the impact of its application in a wide range of agent-based applications, such as supply-chain management [19, 41], roster scheduling [1, 8], meeting scheduling [31], combinatorial auctions [45], bioinformatics [2, 9, 15], and smart home automation [17, 44]. In *Constraint Satisfaction Problems* (CSPs), the goal is to find a value assignment for a set of variables that satisfies a set of constraints [3, 42]. The assignments satisfying the problem constraints are called *solutions*. In *Weighted Constraint Satisfaction Problems* (WCSPs), the goal is that of finding an optimal solution, given a set of preferences expressed by means of cost functions [5, 46, 47].

A key assumption in all these constraint-based models is that *all* the constraints are specified or known a priori. In some applications, such as roster and meeting scheduling problems, some constraints encode the preferences of human users. As such, they may not be fully specified simply because it is unrealistic to accurately know the preferences of users for all possible scenarios in an application. Motivated by such applications, researchers proposed the *Incomplete WCSP* (IWCSP) problem formulation [20], which extends WCSPs by allowing some constraints to be partially specified (i.e., the costs for some constraints are unknown). To solve IWCSPs, they introduced a series of algorithms that interleave the search process, which seeks to find a good solution, and the preference elicitation process, which seeks to obtain some subset of cost functions from the user. Unfortunately, existing approaches suffer from a key limitation – they assume that the elicitation of preferences does not incur any additional cost. This assumption is not realistic as human users are likely bothered by repeated elicitations and will refuse to provide an unbounded number of preferences.

To address this limitation, we make the following contributions:

- We first propose the *IWCSP with Elicitation Costs* (IWCSP+EC) model, which extends the IWCSP model to include the notion of *elicitation costs*. The objective in this problem is to find a solution that minimizes the sum of both the constraint costs and elicitation costs.
- We then introduce three *parameterized* heuristics – Least Unknown Cost (LUC), Least Known Cost (LKC) and their combination heuristic (COM) – that allow users to trade off solution quality for fewer elicited preferences and faster computation times. Further, in settings where elicitations are free (i.e., the elicitation costs are zero), these heuristics also provide *theoretical quality guarantees* on the solutions found.

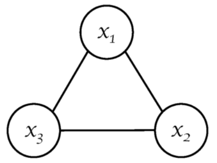
Our experimental results show that COM finds solutions with larger constraint costs than LKC and LUC, but finds them faster and with fewer elicitations than LKC and LUC. Therefore, COM is the preferred heuristic in critical time-sensitive domains. COM also does a better job at trading off solution quality for smaller runtimes, especially when runtimes are large, through the use of user-defined weights. Our model and heuristics thus improve the practical applicability of IWCSPs as they now not only take into account elicitation costs but also provide control knobs, in the form of user-defined weights, to perform tradeoffs along three key dimensions – solution quality, runtime, and number of elicited preferences.

2 BACKGROUND

A *Weighted Constraint Satisfaction Problem* (WCSP) [27, 47] is defined as a tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F} \rangle$:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a finite set of variables;

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.



(a) Constraint Graph

x_1	x_2	$\tilde{f}_1$	f_1	e_1
0	0	15	15	0
0	1	?	11	20
1	0	21	21	0
1	1	?	30	9

x_1	x_3	$\tilde{f}_2$	f_2	e_2
0	0	4	4	0
0	1	?	45	3
1	0	?	8	29
1	1	10	10	0

x_2	x_3	$\tilde{f}_3$	f_3	e_3
0	0	?	50	9
0	1	5	5	0
1	0	7	7	0
1	1	?	6	81

(b) Incomplete Constraint Costs and Elicitation Costs

Figure 1: Example of Incomplete Weighted CSP with Elicitation Costs

- $\mathcal{D} = \{D_1, \ldots, D_n\}$ is a set of finite domains for the variables in $\mathcal{X}$, with D_i being the set of possible values for the variable x_i; and
- $\mathcal{F} = \{f_1, \ldots, f_m\}$ is a set of *weighted constraints* (or *cost tables*). Each weighted constraint is a function $f_i : \bigtimes_{x_j \in \mathbf{x}^{f_i}} D_j \rightarrow \mathbb{R}_0^+ \cup \{\infty\}$, where $\mathbf{x}^{f_i} \subseteq \mathcal{X}$ is the set of variables relevant to f_i, referred to as the *scope* of f_i, and ∞ is a special value denoting that a given combination of value assignments is not allowed.

A *solution* $\mathbf{x}$ is a value assignment to a set of variables $X_{\mathbf{x}} \subseteq \mathcal{X}$ that is consistent with the variables' domains. The cost $\mathbf{F}_{\mathcal{P}}(\mathbf{x}) = \sum_{f \in \mathcal{F}} f(\mathbf{x})$ is the sum of the costs of all the applicable cost functions in $\mathbf{x}$. A solution $\mathbf{x}$ is said to be *complete* if $X_{\mathbf{x}} = \mathcal{X}$. The goal is to find an optimal complete solution $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \mathbf{F}_{\mathcal{P}}(\mathbf{x})$.

An *Incomplete WCSP* (IWCSP) [20] extends WCSPs by allowing some constraints to be *partially specified.* It is defined by a tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \tilde{\mathcal{F}} \rangle$,[1] where $\mathcal{X}$, $\mathcal{D}$, and $\mathcal{F}$ are exactly the same as WCSPs. The key difference is that the set of *fully-specified* constraints $\mathcal{F}$ are not known to an IWCSP algorithm. Instead, only the set of *partially-specified* constraints $\tilde{\mathcal{F}} = \{\tilde{f}_1, \ldots, \tilde{f}_m\}$ are known. Each partially-specified constraint is a function $\tilde{f}_i : \bigtimes_{x_j \in \mathbf{x}^{f_i}} D_j \rightarrow \mathbb{R}_0^+ \cup \{\infty, ?\}$, where ? is a special element denoting that the cost for a given combination of value assignment is not specified. The costs $\mathbb{R}_0^+ \cup \{\infty\}$ that are specified are exactly the costs of the corresponding specified constraints $f_i \in \mathcal{F}$.[2] The goal is still to find an optimal complete solution $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \mathbf{F}_{\mathcal{P}}(\mathbf{x})$, while specifying as few ? of the partially-specified constraints as possible.

3 MOTIVATING APPLICATION

We now briefly describe a *smart home scheduling* problem [17, 44] as a motivating application for our work. In such a problem, an autonomous software agent is deployed in a smart home that is able to automate and schedule the smart IoT devices within the home. To do so effectively, it needs to know the preferences and constraints of users in the home in order to find a schedule that satisfies all the constraints and optimizes the preferences of the users. While some of these preferences may be known, some preferences may be unknown and must be elicited. This problem can be modeled as an IWCSP:

- Variables correspond to smart IoT devices in the home.
- Domains for the variables correspond to the different possible schedules of the devices.
- Weighted constraints correspond to the preferences of users.

[1]The original definition by Gelain et al. [20] does not include the set of weighted constraints $\mathcal{F}$. We include it here to ease the understanding of the different types of solutions.

[2]A constraint $\tilde{f}_i \in \tilde{\mathcal{F}}$ can be fully specified, in which case $\tilde{f}_i = f_i \in \mathcal{F}$.

The objective in this problem is then to find a schedule for each device such that the cumulative preference of the user is optimized while eliciting as few partially-known preferences as possible.

However, in a smart home scheduling problem, users will likely be bothered by the elicitation of their preferences. Therefore, there is a need for a model and approaches that take into account such elicitation costs explicitly. The existing IWCSP model only takes into account such elicitation costs implicitly – through its goal of minimizing the number of preferences elicited during the search. Further, such an assumption also assumes that the elicitation cost is uniform for all preferences, which may be unrealistic. Therefore, we describe in the next section an IWCSP extension that models elicitation costs explicitly and approaches that explicitly take them into account during the search.

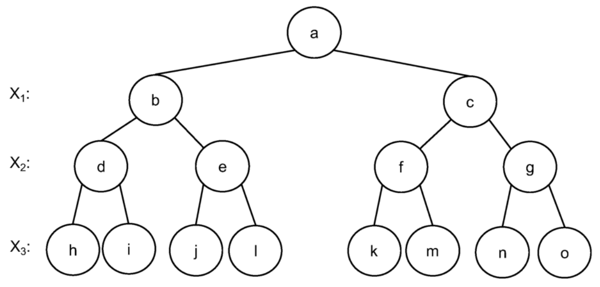
4 IWCSP WITH ELICITATION COSTS

We propose the *IWCSP with Elicitation Cost* (IWCSP+EC) model, which, as the name implies, extends IWCSPs with elicitation costs. It is defined by a tuple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \tilde{\mathcal{F}}, \mathcal{E} \rangle$, where $\mathcal{X}$, $\mathcal{D}$, $\mathcal{F}$, and $\tilde{\mathcal{F}}$ are exactly the same as IWCSPs. $\mathcal{E} = \{e_1, \ldots, e_m\}$ is the set of elicitation costs, where each elicitation cost $e_i : \bigtimes_{x_j \in \mathbf{x}^{f_i}} D_j \rightarrow \mathbb{R}_0^+$ specifies the cost of specifying the weighted constraint f_i (i.e., it is the cost of eliciting the preferences corresponding to constraint f_i).

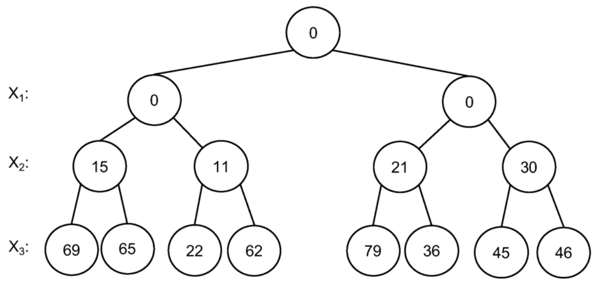
An *explored solution space* $\tilde{\mathbf{x}}$ is the union of all solutions explored thus far by a particular algorithm. The cost $\mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) = \sum_{e \in \mathcal{E}} e(\tilde{\mathbf{x}})$ is the sum of the costs of all applicable elicitation cost functions in $\tilde{\mathbf{x}}$. In other words, it is the cumulative elicitation cost for specifying all ? of partially-specified constraints in the explored solution space.

The total cost $\mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbf{F}_{\mathcal{P}}(\mathbf{x}) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}})$ is the sum of both the cumulative constraint cost $\mathbf{F}_{\mathcal{P}}(\mathbf{x})$ of solution $\mathbf{x}$ and the cumulative elicitation cost $\mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}})$ of explored solutions $\tilde{\mathbf{x}}$. Naturally, the solution $\mathbf{x} \subseteq \tilde{\mathbf{x}}$ must be within the space of explored solutions thus far. The goal is to find an optimal complete solution $\mathbf{x}^*$ while eliciting preferences from a minimal set of solutions $\tilde{\mathbf{x}}^*$ only. More formally, $(\mathbf{x}^*, \tilde{\mathbf{x}}^*) = \operatorname{argmin}_{(\mathbf{x}, \tilde{\mathbf{x}})} \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}})$. We would like to note that it is likely impossible to find such an optimal complete solution *and* elicit preferences related to that solution *only*, except in some very special cases (e.g., only the optimal complete solution has partially-specified constraints).

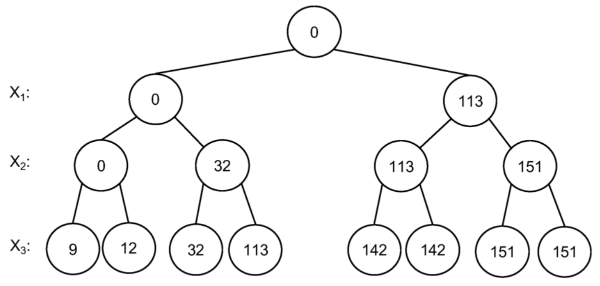
Fig. 1(a) shows the constraint graph of an example IWCSP+EC with three variables x_1, x_2, and x_3, where all variables are constrained with each other. The domains are $D_1 = D_2 = D_3 = \{0, 1\}$. Fig. 1(b) shows both the partially-specified and fully-specified costs as well as the elicitation costs for all constraints. In this example, the optimal complete solution is $\mathbf{x}^* = \langle x_1 = 1, x_2 = 0, x_3 = 1 \rangle$, and only that solution is explored (i.e., $\tilde{\mathbf{x}}^* = \mathbf{x}^*$). The constraint cost



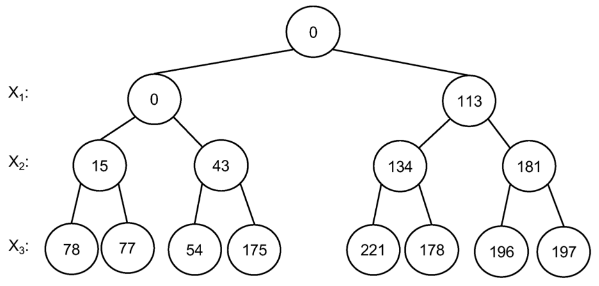
(a) Labels



(b) Constraint Costs



(c) Cumulative Elicitation Costs



(d) Total Costs

Figure 2: Search Trees

of that solution is 36 (= 21 from $f_1(\langle x_1 = 1, x_2 = 0\rangle)$ + 10 from $f_2(\langle x_1 = 1, x_3 = 1\rangle)$ + 5 from $f_3(\langle x_2 = 0, x_3 = 1\rangle)$). As no unknown costs are elicited, the cumulative elicitation cost is 0. Thus, the total cost is 36.

5 DEPTH-FIRST BRANCH-AND-BOUND

Existing IWCSP solvers [20] are all based on the *depth-first branch-and-bound* (DFBnB) search algorithm. As our solvers also use it as the underlying framework, we briefly describe it here. The operations of DFBnB can be visualized with search trees. Fig. 2 shows search trees for our example IWCSP+EC shown in Fig. 1, where levels 1, 2, and 3 correspond to the variables x_1, x_2, and x_3, respectively. Left branches correspond to the variable being assigned the value 0 and right branches correspond to the variable being assigned the value 1. Each non-leaf node thus corresponds to a partial solution of the IWCSP+EC.

Fig. 2(a) shows the identifiers of the nodes that allow us to refer to them easily; Fig. 2(b) shows the constraint costs of the partial solutions if all unknown constraint costs are elicited; Fig. 2(c) shows the cumulative elicitation costs of the explored solution space if one were to expand nodes in a depth-first order and preferring left branches over right branches; and Fig. 2(d) shows the total costs (= sum of constraint and cumulative elicitation costs). For example, node f corresponds to the partial solution $\langle x_1 = 1, x_2 = 0\rangle$ with cost 134 (= constraint cost of 21 + cumulative elicitation cost of 113).

DFBnB expands nodes in the search tree in a depth-first order and prunes nodes whose costs are no smaller than a threshold $\mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}})$, where $\mathbf{x}$ is the best complete solution found so far and $\tilde{\mathbf{x}}$ is the solution space explored thus far. It backtracks once all children of a node have been expanded or pruned.

A simple and straightforward extension of DFBnB to solve IWCSP is as follows, where we use A* notations [23] in some of our definitions: Before expanding a node n, it elicits all the *unknown constraint costs* associated with that node and adds those costs to the *known constraint costs* associated with that node. We refer to these costs as $g(n)$. Additionally, it also adds all the *elicitation costs* associated with the unknown constraint costs elicited to the *cumulative elicitation costs* $\mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}})$, where $\tilde{\mathbf{x}}$ is the union of all partial solutions corresponding to the set of nodes expanded by DFBnB thus far. Finally, one can use heuristics, referred to as $h(n)$, to estimate the sum of constraint and elicitation costs needed to complete the partial solution at node n and if those heuristics are underestimates on the true cost, then they can be used to better prune the search space, that is, when $f(n, \tilde{\mathbf{x}}) = g(n) + h(n) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) \geq \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}})$, where $\mathbf{x}$ is the best complete solution found so far. We provide more formal definitions below, where we use $\mathbf{x}_n$ to refer to the partial solution corresponding to node n.

Definition 5.1 (Unknown Constraint Costs). The unknown constraint costs of a node n are all the partially-specified constraint costs $\tilde{f}(\mathbf{x}) = ?$ that are unknown, where $\mathbf{x} \subseteq \mathbf{x}_n$ is a subset of the partial solution corresponding to node n.

Definition 5.2 (Elicited Constraint Costs). The elicited constraint costs of a node n are all the constraint costs $f(\mathbf{x})$, where $\mathbf{x} \subseteq \mathbf{x}_n$ is a subset of the partial solution corresponding to node n and the partially-specified cost $\tilde{f}(\mathbf{x}) = ?$ of that subset is unknown.

Definition 5.3 (Known Constraint Costs). The known constraint costs of a node n are all the constraint costs $\tilde{f}(\mathbf{x}) \neq ?$, where $\mathbf{x} \subseteq \mathbf{x}_n$ is a subset of the partial solution corresponding to node n.

Definition 5.4 (Cumulative Elicitation Cost). The cumulative elicitation cost of a set of nodes $\tilde{\mathbf{x}}$ is $\mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}})$.

Definition 5.5 (g-Value). The g-value of a node n is $\mathbf{F}_{\mathcal{P}}(\mathbf{x}_n)$.

Definition 5.6 (h-Value). The h-value of a node n is a lower bound estimate of the minimal sum of constraint and elicitation costs to complete the partial solution $\mathbf{x}_n$.

Definition 5.7 (f-Value). The f-value of a node n and the set of nodes $\tilde{\mathbf{x}}$ expanded thus far is the sum of its g-value, h-value, and cumulative elicitation costs.

For example, for node i in Fig. 2(a), its unknown constraint cost is $\tilde{f}_2(\langle x_1 = 0, x_3 = 1\rangle) = ?$; its elicited constraint cost is $f_2(\langle x_1 = 0, x_3 = 1\rangle) = 45$; its known constraint costs are $\tilde{f}_1(\langle x_1 = 0, x_2 = 0\rangle) = 15$ and $\tilde{f}_3(\langle x_2 = 0, x_3 = 1\rangle) = 5$; and its g-value is $g(i) = 45 + 15 + 5 = 65$. If nodes are explored in depth-first order from left to right, then the explored solution space when node i is expanded is $\tilde{\mathbf{x}} = \{\langle x_1 = 0, x_2 = 0, x_3 = 0\rangle, \langle x_1 = 0, x_2 = 0, x_3 = 1\rangle\}$, which includes two unknown constraint costs that were elicited – $f_3(\langle x_2 = 0, x_3 = 0\rangle)$ and $f_2(\langle x_1 = 0, x_3 = 1\rangle)$. The costs of these elicitations are $e_3(\langle x_2 = 0, x_3 = 0\rangle) = 9$ and $e_2(\langle x_1 = 0, x_3 = 1\rangle) = 3$, summing up to a cumulative elicitation cost of 12. Consequently, if we use a zero heuristics (i.e., the h-value for all nodes is 0), then the f-value of node i and the corresponding explored solution space is $f(i, \tilde{\mathbf{x}}) = 65 + 12 = 77$.

6 PARAMETERIZED HEURISTICS

We now describe our three heuristic functions that can be used in conjunction with DFBnB to solve our IWCSP-EC problem. These heuristics make use of an estimated lower bound $\mathcal{L}$ on the cost of all constraints $f \in \mathcal{F}$. Such a lower bound can usually be estimated through domain expertise or can be set to 0 in the worst case since all costs are non-negative. The more informed the lower bound, the more effective the heuristics will be in pruning the search space.

Additionally, these heuristics are parameterized by two parameters – a relative weight $w \geq 1$ and an additive weight $\epsilon \geq 0$. Users can define these parameters a priori allowing them to trade off solution quality for fewer elicited preferences and faster computation times. Further, in settings where elicitations are free (i.e., the elicitation costs are zero), the costs of solutions found are guaranteed to be at most $w \cdot OPT + \epsilon$, where OPT is the optimal solution cost (Theorems 7.1, 7.2, and 7.3).

6.1 Least Unknown Cost (LUC) Heuristic

Our first heuristic function is called the *Least Unknown Cost* (LUC) heuristic. Let S_n be the set of all possible variable-value assignments needed to complete the partial solution corresponding to node n. For each variable-value assignment $\varsigma \in S_n$, let φ_ς denote the number of yet-to-be-elicited unknown constraint costs that must be elicited to complete the partial solution, and E_ς denote the sum of elicitation costs of those constraint costs.

LUC computes the h-value for each node n as follows:

$$h(n) = \min_{\varsigma \in S_n} (\varphi_\varsigma \cdot \mathcal{L} + E_\varsigma) \tag{1}$$

Therefore, the f-value for node n, under the assumption that $\tilde{\mathbf{x}}$ is the set of nodes expanded thus far, is as follows:

$$f(n, \tilde{\mathbf{x}}) = g(n) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) + \min_{\varsigma \in S_n} (\varphi_\varsigma \cdot \mathcal{L} + E_\varsigma) \tag{2}$$

Thus, when using this heuristic, DFBnB prunes a node n if

$$w \cdot f(n, \tilde{\mathbf{x}}) + \epsilon \geq \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}}) \tag{3}$$

where $\mathbf{x}$ is the best complete solution found so far. Note that users can increase the weights w and ϵ, which will prune a larger portion of the search space. Consequently, it will reduce the computation time as well as the number of preferences elicited. However, the downside is that it will also degrade the quality of solutions found.

6.2 Least Known Cost (LKC) Heuristic

Our second heuristic function is called the *Least Known Cost* (LKC) heuristic. Instead of returning h-values like LUC, LKC directly returns the f-values, but taking into account only known and elicited constraint costs. To compute the f-value for each node n, LKC computes the estimated g-values of the leafs in the subtree rooted at n (i.e., all the leaf variables in S_n) – it is an estimate because it does not take into account unknown constraint costs that are yet to be elicited:

$$f(n, \tilde{\mathbf{x}}) = \min_{\varsigma \in S_n} \tilde{g}(l_\varsigma) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) \tag{4}$$

where $\tilde{g}(l_\varsigma)$ is the estimated g-value of node l_ς and l_ς is the leaf node along the branch $\varsigma \in S_n$ that completes the partial solution at node n. Thus, when using this heuristic, DFBnB uses the same pruning condition as Equation 3, except that the f-values are computed using Equation 4.

6.3 Combination (COM) Heuristic

When estimating the minimal cost to complete the partial solution at a node n, the LUC heuristic takes into account yet-to-be-elicited unknown constraint costs and their corresponding elicitation costs. However, it ignores the known and elicited constraint costs needed to complete the partial solution. In contrast, the LKC heuristic does take into account such known and elicited constraint costs. However, it ignores the yet-to-be elicited unknown constraint costs and their corresponding elicitation costs.

Therefore, the LUC and LKC heuristics complement each other as they are estimating non-intersecting components of the minimal cost to complete partial solutions. To take advantage of both heuristics, we combined them into a new *Combination* (COM) heuristic. The new f-values are thus a combination of both the f-values of the LUC and LKC heuristics:

$$f(n, \tilde{\mathbf{x}}) = \min_{\varsigma \in S_n} (\tilde{g}(l_\varsigma) + \varphi_\varsigma \cdot \mathcal{L} + E_\varsigma) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) \tag{5}$$

Thus, when using this heuristic, DFBnB uses the same pruning condition as Equation 3, except that the f-values are computed using Equation 5.

6.4 Value-Ordering Heuristic

Finally, instead of choosing a random order to explore the children of a node, for each heuristic, we order the nodes according to their f-values. DFBnB will then expand the child with the smallest f-value first as that is the most promising child.

7 THEORETICAL RESULTS

We now discuss some theoretical results that are applicable only in the original IWCSP setting, i.e., in problems with zero elicitation costs. Therefore, in this setting, $\mathbf{F}_{\mathcal{P}}(\mathbf{x}) = \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}})$ for all solutions $\mathbf{x}$ and explored search spaces $\tilde{\mathbf{x}}$.

Theorem 7.1. *DFBnB with the LUC heuristic parameterized by a user-defined relative weight $w \geq 1$ and a user-defined additive weight $\epsilon \geq 0$ will return an IWCSP solution whose cost is bounded from above by $w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon$, where $\mathbf{x}^*$ is an optimal complete IWCSP solution.*

Proof. Assume that DFBnB returns a complete solution $\hat{\mathbf{x}}$ with cost $\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}})$. There are the following two cases:

- **Case 1:** $\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) = \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*)$. It is trivial to see that
$$\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) = \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) \leq w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon \tag{6}$$
since $w \geq 1$ and $\epsilon \geq 0$.
- **Case 2:** $\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) > \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*)$. It must be the case that the subtree rooted at some node n along the branch to the optimal solution $\mathbf{x}^*$ was pruned at some point during the search. Otherwise, the algorithm would have returned $\mathbf{x}^*$ since $\mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) < \mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}})$. Therefore, when the subtree was pruned, the following pruning condition from Equation 3 must have held:
$$w \cdot f(n, \tilde{\mathbf{x}}) + \epsilon \geq \mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}}) \tag{7}$$
where $\mathbf{x}$ is the best solution found so far and $\tilde{\mathbf{x}}$ is the search space explored so far. Further, since the RHS of the pruning condition above is non-increasing, it must be the case that:
$$\mathbf{F}_{\mathcal{P}}(\mathbf{x}, \tilde{\mathbf{x}}) \geq \mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) \tag{8}$$
Next, expanding on the LHS of Equation 7:
$$\begin{aligned} w \cdot f(n, \tilde{\mathbf{x}}) + \epsilon &= w \cdot (g(n) + \mathbf{E}_{\mathcal{P}}(\tilde{\mathbf{x}}) + \min_{\varsigma \in S_n} (\varphi_\varsigma \cdot \mathcal{L} + E_\varsigma)) + \epsilon & (9)\\ &= w \cdot (g(n) + \min_{\varsigma \in S_n} \varphi_\varsigma \cdot \mathcal{L}) + \epsilon & (10)\\ &= w \cdot (g(n) + h(n)) + \epsilon & (11)\\ &\leq w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon & (12)\end{aligned}$$
Finally, combining Equations 7 to 12, we get:
$$\mathbf{F}_{\mathcal{P}}(\hat{\mathbf{x}}) \leq w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon \tag{13}$$
which concludes the proof. ■

Theorem 7.2. *DFBnB with the LKC heuristic parameterized by a user-defined relative weight $w \geq 1$ and a user-defined additive weight $\epsilon \geq 0$ will return an IWCSP solution whose cost is bounded from above by $w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon$, where $\mathbf{x}^*$ is an optimal complete IWCSP solution.*

Theorem 7.3. *DFBnB with the COM heuristic parameterized by a user-defined relative weight $w \geq 1$ and a user-defined additive weight $\epsilon \geq 0$ will return an IWCSP solution whose cost is bounded from above by $w \cdot \mathbf{F}_{\mathcal{P}}(\mathbf{x}^*) + \epsilon$, where $\mathbf{x}^*$ is an optimal complete IWCSP solution.*

The proofs for Theorems 7.2 and 7.3 are similar to the proof for Theorem 7.1.

Note that these error bounds do not apply to the IWCSP+EC setting with non-zero elicitation costs. The reason is because the RHS of the pruning condition used by DFBnB is not guaranteed to be non-increasing, thereby invalidating our proof.

8 RELATED WORK

As our work lies in the intersection of constraint-based models, preference elicitation, and heuristic search, we will first focus on related work in this intersection before covering the three broader areas. The body of work that is most related to ours is the work by Gelain et al. [20], where they introduced IWCSPs as well as a family of DFBnB-based algorithms to solve them. They parameterized their algorithm based on *what* preferences must be elicited, *when* the elicitation should take place, and *who* decides the value ordering followed by the algorithm, resulting in 32 combinations of parameter values. They found that the combination that works best is LU.WW.BRANCH. In this combination, preferences with the worst unknown costs are elicited first (what = WW), preferences are elicited once the search reaches a leaf of the search tree (when = BRANCH), and the value ordering is based on a lazy user who prefers values with smaller costs without considering the constraints involving the current variable (who = LU). We compare against this algorithm in our empirical evaluations in the next section. In addition to Incomplete WCSPs, Gelain et al. [20] also introduced *Incomplete Fuzzy CSPs* and generalized both models to *Incomplete Soft CSPs*. Their parameterized algorithms described above are also generalized to solve the general problem.

Another body of work in this intersection is the use of weighted heuristics in *Distributed Constraint Optimization Problems* (DCOPs), which can be viewed as decentralized versions of WCSPs [16, 34, 36, 58]. For example, additive and relative weights were introduced for some algorithms [34, 55, 56], which provide additive and relative quality guarantees as those in Theorem 7.1.

Finally, *Conditional-Preference Networks* (CP-nets) [6, 43] also lie in this intersection. CP-nets are a graphical representation model for qualitative preferences and reflects conditional dependencies between sets of preference statements. In contrast, IWCSPs focuses more on the notion of *conditional additive independence* [4], which requires that the cost of an outcome to be the sum of the "costs" of the different variable values of the outcome.

In the context of the broader constraint-based models where constraints may not be fully specified, there are a number of such models, including *Uncertain CSPs* [59], where the outcomes of constraints are parameterized; *Open CSPs* [14], where the domains of variables and constraints are incrementally discovered; *Dynamic CSPs* [12, 54], where the CSP can change over time; as well as distributed variants of these models [24, 25, 29, 35, 37, 38, 57].

In the context of the broader preference elicitation area, there is a very large body of work [21], and we focus on those that are most closely related to our work. They include techniques that ask users a number of preset questions [49, 51] and send alerts and notification messages to interact with users [11], techniques that ask users to rank alternative options or user-provided option improvements to learn a (possibly approximately) user preference function [7, 10, 50, 53], and techniques that associate costs to eliciting preferences and takes these costs into account when identifying which preference to elicit as well as when to stop eliciting preferences (e.g., when the cost outweighs the expected gain in utility from eliciting any preference) [28, 52]. The key difference between these approaches and ours is that they identify preferences to elicit a priori before the search while we interleave preference elicitation and search.

Finally, in the context of the broader heuristic search area, starting with Weighted A* [39], researchers have long used weighted heuristics to speed up the search process in general search problems. Often, solutions found by these weighted approaches also have similar quality guarantees as those in Theorem 7.1. Researchers have also investigated the use of dynamically-changing weights [40, 48]; using weighted heuristic with other heuristic search algorithms like DFBnB [18], RBFS [26], and AND/OR search [32, 33]; as well as extending them to provide anytime characteristics [22, 30].

9 EXPERIMENTAL RESULTS

We evaluate DFBnB using our three heuristics – LUC, LKC, and COM – against a random (RND) heuristic on both IWCSP+ECs and IWCSPs (without elicitation costs). For IWCSPs, we also compare against the best algorithm proposed by Gelain et al. [20] – the LU.WW.BRANCH algorithm (labeled LWB). We evaluate the algorithms on random graphs, where we measure the various costs of the solutions found – the cumulative constraint costs, cumulative elicitation costs, and their aggregated total cost – the number of unknown costs elicited to find those solutions, and the runtime of the algorithms. All experiments were performed on an Intel Core i7, 3.4GHz machine with 16GB of RAM. Each data point shown is an average of over 100 instances.

We generate 100 random (binary) graphs [13], where we vary the number of variables $|\mathcal{X}|$ from 5 to 12; the constraint density p_1 from 0.2 to 0.8; the fraction of unknown costs i in each constraint from 0.2 to 1.0;[3]. the user-defined relative weight w from 1 to 10; and the user-defined additive weight ϵ from 0 to 1000. The domain size $|D_i|$ for all variables $x_i \in \mathcal{X}$ is set to 3. In our experiments below, we only vary one parameter at a time, setting the rest at their default values: $|\mathcal{X}| = 10$, $p_1 = 0.4$, $i = 0.6$, $w = 1$, and $\epsilon = 0$. All constraint costs are randomly sampled from $[2, 100]$ and all elicitation costs are randomly sampled from $[0, 20]$.

IWCSP+ECs: Table 1 tabulates the empirical results for our IWCSP+EC experiments, where we vary number of variables $|\mathcal{X}|$, the density p_1, and the fraction of unknown costs i. We make the following observations with regards to the runtime of the algorithms and the number of unknown costs that they elicit:

- As expected, the runtimes of all algorithms increase with increasing number of variables $|\mathcal{X}|$ and constraint density p_1. This trend is also reflected in the number of unknown costs elicited. The reason is that the size of the problem, in terms of the number of constraints in the problem, increases with increasing $|\mathcal{X}|$ and p_1. And all algorithms need to elicit more unknown costs and evaluate the costs of more constraints before terminating.
- Interestingly, while the number of unknown costs elicited also increases for all algorithms with increasing fraction of unknown costs i, the runtimes of LKC and LUC decrease instead. The reason is that the ratio of unknown costs elicited compared to the total number of unknown costs actually decreases with increasing i. Since the size of the problem, in terms of the number of constraints in the problem, is the same for all values of i (as they are dependent only on $|\mathcal{X}|$ and p_1, which remain unchanged), the ratio of unknown costs elicited roughly reflects the ratio of search space explored. Therefore, as this ratio decreases with i, so does the runtime of LKC and LUC.

  The exception to this observation is COM, whose runtimes increase slightly from when $i = 0.2$ to $i = 0.4$, after which it plateaus. However, the reason is similar to the one above – the ratio of unknown costs elicited compared to the total number of unknown costs increased slightly when $i = 0.2$ to $i = 0.4$ and remained relatively unchanged for larger values of i.
- Finally, COM is generally faster than both LKC and LUC across all parameters. The reason is because COM is able to prune a larger portion of the search space compared to LKC and LUC, as reflected by the observation that the number of unknown costs elicited by COM is smaller than that of LKC and LUC.

We now discuss the costs of the solutions found in terms of their cumulative constraint costs, cumulative elicitation costs, and their aggregated total costs:

- As expected, the cumulative elicitation cost of the solutions found, which is proportional to the number of unknown costs elicited, by all algorithms increase with increasing number of variables $|\mathcal{X}|$, constraint density p_1, and fraction of unknown costs i. The reason is that the number of unknown costs in the problem increases with increasing $|\mathcal{X}|$, p_1, and i.
- The cumulative constraint cost of the solutions found by all algorithms also increase with increasing $|\mathcal{X}|$, p_1, and i. However, the reasons for why they increase is different: The constraint costs increase for increasing $|\mathcal{X}|$ and p_1 because the number of constraints in the problem increases. As such, the average constraint cost of solutions increases with increasing $|\mathcal{X}|$ and p_1.

  In contrast, the average constraint cost of solutions should remain relatively unchanged with increasing i because the number of constraints remain unchanged. In this case, the constraint costs of solutions found increase because the algorithms find increasingly worse solutions with increasing i. As the algorithms do not know the unknown constraint costs until they are elicited, they will prefer to elicit unknown costs that have smaller elicitation costs. And since the elicitation cost and unknown constraint cost are not correlated, the behavior of the algorithms become increasingly random with increasing i, thereby resulting in solutions with larger constraint costs.
- Finally, since both the cumulative elicitation costs and cumulative constraint costs increase with increasing $|\mathcal{X}|$, p_1, and i, the aggregated total costs also increase similarly.
- In general, COM finds solutions with larger constraint costs than LKC and LUC. However, its elicitation costs are smaller than those of LKC and LUC. (Our results are statistically significant with $p < 0.05$). This implies that COM is able to find relatively good solutions quickly and uses that solution to prune a large portion of the search space. In contrast, LKC and LUC explores a larger portion of the search space to find better solutions, but at the cost of increasing elicitation cost. This behavior is also reflected in the runtimes of COM, which are smaller than the runtimes of LKC and LUC. The total cost of solutions found by all three algorithms are all approximately the same.

Fig. 3 plots the empirical results for our IWCSP+EC experiments, where we vary the relative weight w from 1 to 10 in increments

[3]In other words, for each constraint $\tilde{f} \in \tilde{\mathcal{F}}$, i is the number of value assignments with unspecified costs as a fraction of the total number of value assignments in that constraint.

(a) Varying Number of Variables $|\mathcal{X}|$, $i = 0.6$, $p_1 = 0.4$

$\|\mathcal{X}\|$	# unknown	# of elicited costs			runtime (sec)			total cost			cumulative constraint cost			cumulative elicitation cost		
	costs	LKC	LUC	COM	LKC	LUC	COM	LKC	LUC	COM	LKC	LUC	COM	LKC	LUC	COM
5	20.75	14.32	12.16	11.83	0.03	0.03	0.03	167.93	151.50	160.30	98.41	115.97	109.45	65.37	53.09	50.85
6	30.60	21.71	18.23	17.06	0.11	0.10	0.09	264.17	241.08	248.97	171.97	161.85	177.42	92.20	79.23	71.55
7	40.90	30.44	26.51	23.31	0.39	0.33	0.34	366.68	339.77	354.28	242.34	243.01	260.36	124.34	96.76	93.92
8	55.30	41.73	36.39	30.31	1.63	1.64	1.35	515.34	480.56	483.67	352.83	347.82	365.52	162.51	132.74	118.15
9	70.00	53.16	46.26	39.27	7.74	7.26	5.35	673.10	630.47	637.40	485.87	467.85	499.14	187.23	162.62	138.26
10	90.05	68.24	60.52	52.01	34.11	33.73	20.96	867.03	833.77	861.34	646.32	656.12	688.39	220.71	177.65	172.95
11	110.05	84.39	75.77	62.77	192.98	182.93	101.12	1086.84	1047.23	1058.42	845.08	831.48	872.95	241.76	215.75	185.47
12	130.00	100.41	89.31	73.95	1505.35	1301.48	552.12	1295.68	1248.14	1258.51	1003.53	1014.30	1047.98	292.15	233.84	210.53

(b) Varying Constraint Density p_1, $|\mathcal{X}| = 10$, $i = 0.6$

p_1	# unknown	# of elicited costs			runtime (sec)			total cost			cumulative constraint cost			cumulative elicitation cost		
	costs	LKC	LUC	COM	LKC	LUC	COM	LKC	LUC	COM	LKC	LUC	COM	LKC	LUC	COM
0.2	56.70	44.67	38.92	33.15	34.08	26.56	23.15	546.26	501.72	505.75	365.32	359.93	374.56	180.94	141.79	131.19
0.4	90.05	68.24	60.52	52.01	34.11	33.73	20.96	867.03	833.77	861.34	646.32	656.12	688.39	220.71	177.65	172.95
0.6	135.00	97.74	89.41	74.15	34.36	40.78	27.35	1333.73	1331.18	1335.97	1053.82	1086.79	1116.19	279.91	244.39	219.78
0.8	180.00	124.55	117.58	92.29	37.59	49.13	33.57	1813.18	1806.55	1817.44	1493.86	1522.51	1534.46	319.32	284.04	282.98

(c) Varying Fraction of Unknown Costs i, $|\mathcal{X}| = 10$, $p_1 = 0.4$

i	# unknown	# of elicited costs			runtime (sec)			total cost			cumulative constraint cost			cumulative elicitation cost		
	costs	LKC	LUC	COM	LKC	LUC	COM	LKC	LUC	COM	LKC	LUC	COM	LKC	LUC	COM
0.2	36.02	30.15	25.40	13.71	48.58	41.77	16.97	666.87	662.86	718.70	544.06	563.76	645.59	122.81	99.10	73.11
0.4	72.04	57.75	49.63	37.14	43.22	36.15	19.40	809.65	790.16	802.20	615.10	637.36	665.96	194.55	152.80	136.24
0.6	90.05	68.24	60.52	52.01	34.11	33.73	20.96	867.03	833.77	861.34	646.32	656.12	688.39	220.71	177.65	172.95
0.8	126.07	85.40	80.64	73.95	25.09	28.14	22.66	964.39	941.53	928.84	712.97	712.14	707.86	251.42	229.39	220.98
1.0	162.09	90.34	91.56	82.16	20.51	23.70	21.79	1019.10	1019.87	1018.80	772.25	775.46	768.83	246.85	244.41	249.97

Table 1: IWCSP+EC Empirical Results for Random Graphs

of 1 and the additive weight ϵ from 0 to 1000 in increments of 100. Each data point in the figures thus shows the result for one of the heuristic with one of the values of w or ϵ. Data points for smaller values of w and ϵ are in the bottom right of the figures and data points for larger values are in the top left of the figures.

We plot the tradeoffs between total costs (= cumulative constraint and elicitation costs) and number of elicited costs as well as the tradeoffs between total costs and runtimes. As expected, as the relative and additive weights increase, the total costs increase, the number of elicited costs decreases, and the runtimes decrease. The key difference between both weights is that the tradeoffs are much more uniform when using additive weights compared to relative weights. Therefore, using additive weights may allow users to better control the granularity of the tradeoffs. Compared to our three heuristics, the random heuristic performs poorly and randomly.

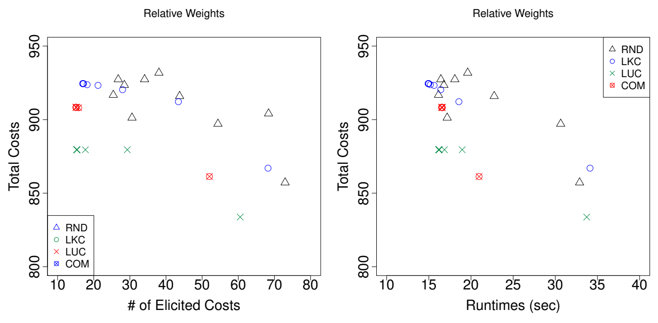
When comparing the tradeoffs between total costs and number of elicited costs, LUC is better than COM, which is better than LKC. In other words, to find solutions of the same cost, LUC elicits fewer costs than COM, which elicits fewer costs than LKC. However, when comparing the tradeoffs between total costs and runtimes, the same trend applies when the runtimes are small, but COM is better than LUC when the runtimes are large (larger than 20s in our experiments). Therefore, when the weights are sufficiently large, COM provides the best tradeoff between total costs and runtimes.

IWCSPs: We now present our empirical results for our IWCSP experiments. Note that these are problems without elicitation costs and the weights define the theoretical error bounds on the quality of solutions found in these problems (Theorems 7.1, 7.2, and 7.3).

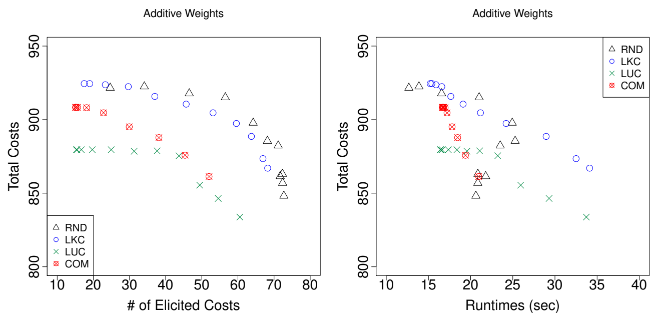
While the original LWB algorithm does not allow users to define error bounds, it could be extended to do so since it uses its own specific heuristic function. We thus parameterize it the same way using Equation 3 and compare against it in our experiments below.

Fig. 4 plots the empirical results, where we vary the relative weight w and additive weight ϵ. The key difference for these plots compared to the earlier ones is that the results for LWB is also included. We make the following observations:

- Similar to the trends in IWCSP+EC problems, as the additive and relative weights increase (from the bottom right of the figures to the top left of the figures), the total costs of solutions increase, the number of elicited costs decreases, and the runtimes decrease. Additionally, the random heuristic also performs poorly in this setting as expected.
- When comparing the tradeoffs between total costs and number of elicited costs, COM and LKC behave similarly, and they are both better than LUC. Interestingly, this trend is the opposite of that in IWCSP+EC problems, where LUC was the best. The reason is that the heuristic function used by LUC to estimate the cost to complete the current partial solution is poor when elicitation costs are not taken into account. In such a case, the only contribution to the estimate is the number of yet-to-be elicited unknown costs, which is then multiplied by a lower bound $\mathcal{L}$ on the cost of all constraints. In our experiments, $\mathcal{L} = 2$, which is a poor estimate as the costs can be as large as 100.
- This trend is also repeated when comparing the tradeoffs between total costs and runtimes – to find solutions of the same cost, COM and LKC requires a smaller runtime than LUC.

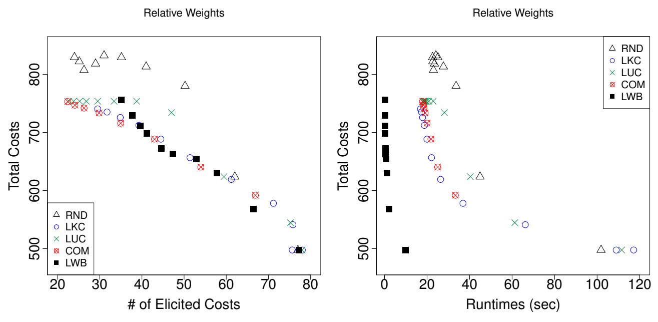


(a) Varying Relative Weight w

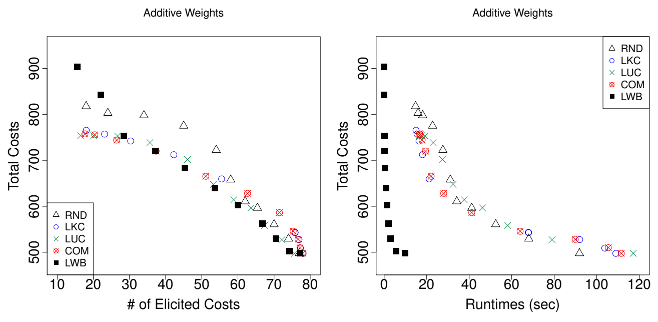


(b) Varying Additive Weight ϵ

Figure 3: IWCSP+EC Empirical Results for Random Graphs with $|\mathcal{X}| = 10$, $p_1 = 0.4$, and $i = 0.6$



(a) Varying Relative Weight w



(b) Varying Additive Weight ϵ

Figure 4: IWCSP Empirical Results for Random Graphs with $|\mathcal{X}| = 10$, $p_1 = 0.4$, and $i = 0.6$

- When searching for optimal solutions (i.e., $w = 1$ and $\epsilon = 0$), all three of our algorithms elicit similar number of costs compared to LWB. LWB is also about one order of magnitude faster than our three algorithms. The reason is because, unlike our algorithms, LWB does not use any heuristics to estimate the cost to complete the solution.
- When searching for suboptimal solutions (i.e., $w > 1$ or $\epsilon > 0$), all three of our algorithms find solutions of similar costs compared to LWB but with smaller number of elicited costs once w is sufficiently large. They also find solutions with smaller costs but with similar number of elicited costs once ϵ is sufficiently large.

10 CONCLUSIONS

Incomplete Weighted Constraint Satisfaction Problems (IWCSPs) are an elegant paradigm for modeling combinatorial optimization problems with partially-specified constraints. To fully specify such constraints, one must elicit preferences of users. Unfortunately, existing IWCSP approaches assume that the elicitation of preferences does not incur any additional cost. This is unrealistic as human users are likely bothered by repeated elicitations and will refuse to provide an unbounded number of preferences.

To overcome this limitation, we proposed the *IWCSP with Elicitation Costs* (IWCSP+EC) model, which extends the IWCSP model to include the notion of *elicitation costs*. The objective in this problem is to find a solution that minimizes the sum of both the constraint costs and elicitation costs. We also introduced three *parameterized* heuristics – Least Unknown Cost (LUC), Least Known Cost (LKC) and their combination heuristic (COM) – that allow users to trade off solution quality for fewer elicited preferences and faster computation times. Further, in settings where elicitations are free, these heuristics also provide *theoretical quality guarantees* on the solutions found.

Our empirical results show that COM finds solutions with larger constraint costs than LKC and LUC, but finds them faster and with fewer elicitations than LKC and LUC. Therefore, COM is the preferred heuristic in critical time-sensitive domains. COM also does a better job at trading off solution quality for smaller runtimes, especially when runtimes are large, through the use of user-defined weights. In conclusion, our heuristics improve the practical applicability of IWCSPs as they now not only take into account elicitation costs but also provide control knobs, in the form of user-defined weights, to perform tradeoffs along three key dimensions – solution quality, runtime, and number of elicited preferences.

ACKNOWLEDGMENTS

Tabakhi and Yeoh are partially supported by NSF grants 1550662 and 1812619 and Yokoo is partially supported by JSPS KAKENHI grant 17H00761.

REFERENCES

[1] Slim Abdennadher and Hans Schlenker. 1999. Nurse Scheduling using Constraint Logic Programming. In *Proceedings of IAAI*. 838–843.

[2] David Allouche, Isabelle André, Sophie Barbe, Jessica Davies, Simon de Givry, George Katsirelos, Barry O'Sullivan, Steven David Prestwich, Thomas Schiex, and Seydou Traoré. 2014. Computational protein design as an optimization problem. *Artificial Intelligence* 212 (2014), 59–79.

[3] Krzysztof Apt. 2003. *Principles of Constraint Programming*. Cambridge University Press.

[4] Fahiem Bacchus and Adam J. Grove. 1996. Utility Independence in a Qualitative Decision Theory. In *Proceedings of KR*. 542–552.

[5] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. 1997. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44, 2 (1997), 201–236.

[6] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. 2004. CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *Journal of Artificial Intelligence Research* 21 (2004), 135–191.

[7] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. 2006. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence* 170, 8-9 (2006), 686–713.

[8] Edmund K Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. 2004. The state of the art of nurse rostering. *Journal of Scheduling* 7, 6 (2004), 441–499.

[9] Federico Campeotto, Alessandro Dal Palù, Agostino Dovier, Ferdinando Fioretto, and Enrico Pontelli. 2013. A Constraint Solver for Flexible Protein Model. *Journal of Artificial Intelligence Research* 48 (2013), 953–1000.

[10] Urszula Chajewska, Daphne Koller, and Ronald Parr. 2000. Making Rational Decisions Using Adaptive Utility Elicitation. In *Proceedings of AAAI*. 363–369.

[11] Enrico Costanza, Joel E. Fischer, James A. Colley, Tom Rodden, Sarvapali D. Ramchurn, and Nicholas R. Jennings. 2014. Doing the laundry with agents: a field trial of a future smart energy system in the home. In *Proceedings of CHI*. 813–822.

[12] Rina Dechter and Avi Dechter. 1988. Belief Maintenance in Dynamic Constraint Networks. In *Proceedings of AAAI*. 37–42.

[13] Paul Erdös and Alfréd Rényi. 1959. On random graphs, I. *Publicationes Mathematicae (Debrecen)* 6 (1959), 290–297.

[14] Boi Faltings and Santiago Macho-Gonzalez. 2005. Open constraint programming. *Artificial Intelligence* 161, 1-2 (2005), 181–208.

[15] Ferdinando Fioretto, Agostino Dovier, and Enrico Pontelli. 2015. Constrained Community-Based Gene Regulatory Network Inference. *ACM Transactions on Modeling and Computer Simulation* 25, 2 (2015), 11:1–11:26.

[16] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. 2018. Distributed Constraint Optimization Problems and Applications: A Survey. *Journal of Artificial Intelligence Research* 61 (2018), 623–698.

[17] Ferdinando Fioretto, William Yeoh, and Enrico Pontelli. 2017. A Multiagent System Approach to Scheduling Devices in Smart Homes. In *Proceedings of AAMAS* . 981–989.

[18] Natalia Flerova, Radu Marinescu, and Rina Dechter. 2017. Weighted heuristic anytime search: new schemes for optimization over graphical models. *Annals of Mathematics and Artificial Intelligence* 79, 1-3 (2017), 77–128.

[19] Jonathan Gaudreault, Jean-Marc Frayret, and Gilles Pesant. 2009. Distributed search for supply chain coordination. *Computers in Industry* 60, 6 (2009), 441–451.

[20] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, K Brent Venable, and Toby Walsh. 2010. Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. *Artificial Intelligence* 174, 3-4 (2010), 270–294.

[21] Judy Goldsmith and Ulrich Junker. 2008. Preference Handling for Artificial Intelligence. *AI Magazine* 29, 4 (2008), 9–12.

[22] Eric Hansen and Rong Zhou. 2007. Anytime Heuristic Search. *Journal of Artificial Intelligence Research* 28.

[23] Paul Hart, Nils Nilsson, and Bertram Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

[24] Khoi D. Hoang, Ferdinando Fioretto, Ping Hou, Makoto Yokoo, William Yeoh, and Roie Zivan. 2016. Proactive Dynamic Distributed Constraint Optimization. In *Proceedings of AAMAS*. 597–605.

[25] Khoi D. Hoang, Ping Hou, Ferdinando Fioretto, William Yeoh, Roie Zivan, and Makoto Yokoo. 2017. infinite-Horizon Proactive Dynamic DCOPs. In *Proceedings of AAMAS*. 212–220.

[26] Richard E Korf. 1993. Linear-space best-first search. *Artificial Intelligence* 62, 1 (1993), 41–78.

[27] Javier Larrosa. 2002. Node and arc consistency in weighted CSP. In *Proceedings of AAAI/IAAI*. 48–53.

[28] Tiep Le, Atena M. Tabakhi, Long Tran-Thanh, William Yeoh, and Tran Cao Son. 2018. Preference Elicitation with Interdependency and User Bother Cost. In *Proceedings of AAMAS*. 1459–1467.

[29] Thomas Léauté and Boi Faltings. 2011. Distributed Constraint Optimization under Stochastic Uncertainty. In *Proceedings of AAAI*. 68–73.

[30] Maxim Likhachev, Geoffrey Gordon, and Sebastian Thrun. 2003. ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In *Proceedings of NIPS*.

[31] Rajiv Maheswaran, Milind Tambe, Emma Bowring, Jonathan Pearce, and Pradeep Varakantham. 2004. Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Event Scheduling. In *Proceedings of AAMAS*. 310–317.

[32] Radu Marinescu and Rina Dechter. 2005. AND/OR branch-and-bound for graphical models. In *Proceedings of IJCAI*. 224–229.

[33] Radu Marinescu and Rina Dechter. 2009. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence* 173, 16-17 (2009), 1457–1491.

[34] Pragnesh Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. 2005. ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees. *Artificial Intelligence* 161, 1–2 (2005), 149–180.

[35] Duc Thien Nguyen, William Yeoh, and Hoong Chuin Lau. 2012. Stochastic Dominance in Stochastic DCOPs for Risk-Sensitive Applications. In *Proceedings of AAMAS*. 257–264.

[36] Adrian Petcu and Boi Faltings. 2005. A Scalable Method for Multiagent Constraint Optimization. In *Proceedings of IJCAI*. 1413–1420.

[37] Adrian Petcu and Boi Faltings. 2005. Superstabilizing, Fault-Containing Multiagent Combinatorial Optimization. In *Proceedings of AAAI*. 449–454.

[38] Adrian Petcu and Boi Faltings. 2006. ODPOP: An Algorithm for Open/Distributed Constraint Optimization. In *Proceedings of AAAI*. 703–708.

[39] Ira Pohl. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence* 1, 3 (1970), 193–204.

[40] Ira Pohl. 1973. The Avoidance of (Relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving. In *Proceedings of IJCAI*. 12–17.

[41] L.C.A. Rodrigues and L. Magatao. 2007. Enhancing Supply Chain Decisions Using Constraint Programming: A Case Study. In *Proceedings of MICAI*. 1110–1121.

[42] F. Rossi, P. van Beek, and T. Walsh (Eds.). 2006. *Handbook of Constraint Programming*. Elsevier.

[43] Francesca Rossi, Kristen Brent Venable, and Toby Walsh. 2008. Preferences in Constraint Satisfaction and Optimization. *AI Magazine* 29, 4 (2008), 58–68.

[44] Pierre Rust, Gauthier Picard, and Fano Ramparany. 2016. Using Message-Passing DCOP Algorithms to Solve Energy-Efficient Smart Environment Configuration Problems. In *Proceedings of IJCAI*. 468–474.

[45] Tuomas Sandholm. 2002. Algorithm for Optimal Winner Determination in Combinatorial Auctions. *Artificial Intelligence* 135, 1–2 (2002), 1–54.

[46] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. 1995. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proceedings of IJCAI*. 631–637.

[47] Linda G Shapiro and Robert M Haralick. 1981. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 3, 5 (1981), 504–519.

[48] Xiaoxun Sun, Marek Druzdzel, and Changhe Yuan. 2007. Dynamic Weighting A* Search-based MAP Algorithm for Bayesian Networks. In *Proceedings of IJCAI*. 2385–2390.

[49] Atena M. Tabakhi, Tiep Le, Ferdinando Fioretto, and William Yeoh. 2017. Preference Elicitation for DCOPs. In *Proceedings of CP*. 278–296.

[50] Stefano Teso, Paolo Dragone, and Andrea Passerini. 2017. Coactive Critiquing: Elicitation of Preferences and Features. In *Proceedings of AAAI*. 2639–2645.

[51] Walid Trabelsi, Kenneth N. Brown, and Barry O'Sullivan. 2015. Preference Elicitation and Reasoning While Smart Shifting of Home Appliances. *Energy Procedia* 83 (2015), 389–398.

[52] Ngoc Cuong Truong, Tim Baarslag, Sarvapali D. Ramchurn, and Long Tran-Thanh. 2016. Interactive Scheduling of Appliance Usage in the Home. In *Proceedings of IJCAI*. 869–877.

[53] Paolo Viappiani and Craig Boutilier. 2010. Optimal Bayesian Recommendation Sets and Myopically Optimal Choice Query Sets. In *Proceedings of NIPS*. 2352–2360.

[54] Richard Wallace and Eugene Freuder. 1998. Stable Solutions for Dynamic Constraint Satisfaction Problems. In *Proceedings of CP*. 447–461.

[55] William Yeoh, Ariel Felner, and Sven Koenig. 2010. BnB-ADOPT: An Asynchronous Branch-and-Bound DCOP Algorithm. *Journal of Artificial Intelligence Research* 38 (2010), 85–133.

[56] William Yeoh, Xiaoxun Sun, and Sven Koenig. 2009. Trading Off Solution Quality for Faster Computation in DCOP Search Algorithms. In *Proceedings of IJCAI*. 354–360.

[57] William Yeoh, Pradeep Varakantham, Xiaoxun Sun, and Sven Koenig. 2015. Incremental DCOP Search Algorithms for Solving Dynamic DCOPs. In *Proceedings of IAT*. 257–264.

[58] William Yeoh and Makoto Yokoo. 2012. Distributed Problem Solving. *AI Magazine* 33, 3 (2012), 53–65.

[59] Neil Yorke-Smith and Carmen Gervet. 2003. Certainty Closure: A Framework for Reliable Constraint Reasoning with Uncertainty. In *Proceedings of CP*. 769–783.