Efficient Satisfiability Verification for Conditional Importance Networks

Zachary J. Oster
University of Wisconsin-Whitewater
Whitewater, Wisconsin, USA
osterz@uww.edu

Abstract

Conditional importance networks (CI-nets) provide a formal framework for modeling and reasoning with qualitative preferences over sets of many variables. Existing approaches for verifying the satisfiability of a CI-net operate on a complete model of the CI-net's semantics, but the time required to construct this model is exponential in the number of preference variables; this inefficiency limits the practical usefulness of CI-nets. To bypass this performance bottleneck, we present a new algorithm that decides whether a CI-net is satisfiable by analyzing a sufficient partial model of its semantics, and we show how to efficiently construct such a partial model. Our approach can significantly reduce the average time required to verify a CI-net's satisfiability compared to existing methods.

1 Introduction

Conditional importance networks, or CI-nets [Bouveret *et al.*, 2009], provide an expressive language for specifying, modeling, and reasoning over qualitative preferences among sets of items. Preferences of this type appear in a wide variety of problem domains, including software requirements engineering [Oster *et al.*, 2015], online trust negotiation [Oster *et al.*, 2012], and cybersecurity [Santhanam *et al.*, 2013b].

In order to guarantee the correctness of conclusions that are reached by reasoning over a CI-net, it is necessary to verify that the CI-net is *satisfiable*. Existing methods for verifying CI-net satisfiability, including those defined by Bouveret *et al.* [2009] and Santhanam *et al.* [2010], begin by constructing a graphical model called a *preference graph* that explicitly represents the (partial) ordering of preferences encoded in the semantics of the given CI-net. Once the preference graph is constructed, the next step is to check the graph for cycles; the CI-net is satisfiable if and only if its preference graph is acyclic [Bouveret *et al.*, 2009]. Unfortunately, constructing the preference graph is a computationally intensive task that requires $\Theta(2^N)$ time for a CI-net that expresses preferences over N variables (preference items) [Oster *et al.*, 2012].

The main contribution of this paper is a novel algorithm for verifying CI-net satisfiability without needing to construct and analyze the entire preference graph. Since the complexity of CI-net satisfiability checking is dominated by the time and memory costs of constructing the semantic model to be analyzed, our approach can significantly reduce the time and memory required for satisfiability checking compared to existing approaches. In the process, we prove several results about the semantics of CI-nets, including a new satisfiability condition that provides the theoretical basis for our verification algorithm's correctness. We also describe our implementation of this new algorithm and provide an empirical evaluation based on that implementation.

The remainder of the paper is organized as follows. Section 2 gives a short overview of CI-nets and the CI-net satisfiability checking problem. Section 3 presents our novel approach to CI-net satisfiability checking, including proofs of correctness. Section 4 summarizes the results from our empirical evaluation of this satisfiability checking method. Finally, Section 5 discusses future applications of this work.

2 Overview of CI-Nets

Conditional importance networks (CI-nets), which were first introduced by Bouveret *et al.* [2009], allow stakeholders to specify relative importance among sets of items, conditioned on the presence or absence of other items.

2.1 Syntax and Semantics

Let $\mathbf{V}$ be a finite set of binary variables, both here and throughout this paper. We will use the definitions from [Bouveret *et al.*, 2009] for conditional importance statements (CI-statements) and conditional importance networks (CI-nets), which are given here as Definitions 1 and 2.

Definition 1 *A* conditional importance statement *(*CI-statement*) on $\mathbf{V}$ is a quadruple (S^+, S^-, S_1, S_2) of pairwise disjoint subsets of $\mathbf{V}$. This CI-statement can be written as $S^+, S^- : S_1 \succ S_2$.*

Informally, a CI-statement can be interpreted to mean: "Given two sets of items from $\mathbf{V}$, if both sets include the items in S^+ and neither set includes the items in S^-, then I would rather have the set that has all items in S_1 instead of the set that has all items in S_2, *ceteris paribus* (all else being equal)."

Definition 2 *A CI-net on $\mathbf{V}$ is a set $\mathbf{C}$ of CI-statements on $\mathbf{V}$.*

Formally, a CI-net $\mathbf{C}$ over a set of variables $\mathbf{V}$ is *satisfiable* if and only if there exists a strict partial order (irreflexive, asymmetric, and transitive) relation $\succ$ over the powerset of $\mathbf{V}$ such that:

1. For each CI-net statement $S^+, S^- : S_1 \succ S_2$, if $\bar{S} \subseteq \mathbf{V} \setminus (S^+ \cup S^- \cup S_1 \cup S_2)$ then $\bar{S} \cup S^+ \cup S_1 \succ \bar{S} \cup S^+ \cup S_2$.
2. $\succ$ is monotonic (for any $V_i, V_j \subseteq \mathbf{V}$, $V_i \supset V_j \Rightarrow V_i \succ V_j$).

Bouveret *et al.* [2009] define the semantics of CI-statements in terms of *worsening flips*, where each worsening flip is a pair of sets of variables (V_1, V_2) (where $V_1, V_2 \subseteq \mathbf{V}$) such that V_1 is preferred to V_2 ($V_1 \succ V_2$); i.e., replacing the set V_1 with V_2 results in a worse outcome. In contrast, Santhanam *et al.* [2010] define the CI-statement semantics in terms of *improving flips*, where the same pair (V_1, V_2) indicates that V_2 is preferred to V_1 ($V_2 \succ V_1$); i.e., replacing the set V_1 with V_2 results in a better outcome.

It is important to note that all results proven in this paper hold regardless of whether worsening flips or improving flips are used. From this point on, the word "flip" will denote either a worsening or an improving flip unless specifically noted. Although several of our definitions and proofs are based on the use of improving flips, it is straightforward to rewrite any of these using worsening flips.

Definition 3 *([Santhanam* et al.*, 2010], after [Bouveret* et al.*, 2009]) A sequence of sets of variables $V_1, V_2, \cdots, V_{n-1}, V_n$ is an* improving flipping sequence *with respect to a set of CI-net statements if and only if, for $1 \leq i < n$, either*

1. *(Monotonicity Flip) $V_{i+1} \supset V_i$; or*
2. *(Importance Flip) there exists a conditional importance statement $S^+, S^- : S_1 \succ S_2$ in the CI-net, such that all three of the following conditions are satisfied:*
   (a) $V_{i+1} \supseteq S^+$, $V_i \supseteq S^+$, and $V_{i+1} \cap S^- = V_i \cap S^- = \emptyset$;
   (b) $V_{i+1} \supseteq S_1$, $V_i \supseteq S_2$, and $V_{i+1} \cap S_2 = V_i \cap S_1 = \emptyset$;
   (c) if $V = V \setminus (S^+ \cup S^- \cup S_1 \cup S_2)$, then $V \cap S_1 = V \cap S_2$.

In this definition, condition 1 states that a set containing more variables is always preferred to a set containing fewer variables. Condition 2 states that if the variables in S^+ are included and the variables in S^- are not included, then including the variables in S_1 is more important than including the variables in S_2, all else being equal (*ceteris paribus*). Given a CI-net $\mathbf{C}$ and two variable sets V_1 and V_2, we say that V_1 is preferred to V_2 under $\mathbf{C}$, denoted by $\mathbf{C} \models V_1 \succ V_2$, if and only if there is an improving flipping sequence with respect to $\mathbf{C}$ from V_2 to V_1 (Proposition 1 in [Bouveret *et al.*, 2009]).

From Definition 3, one can construct a graph where each node corresponds to a set of variables and each directed edge from one node to another denotes a flip. This graph is called the *preference graph* of $\mathbf{C}$ in [Bouveret *et al.*, 2009] and the *induced preference graph* of $\mathbf{C}$ in [Santhanam *et al.*, 2010; 2013a; Oster *et al.*, 2015]; we use the shorter name here.

Definition 4 *([Oster* et al.*, 2015]) Given a CI-net $\boldsymbol{C}$ over a set of variables $\mathbf{V}$, the* preference graph $G(\boldsymbol{C}) = (N, E)$ *is constructed as follows. The nodes N correspond to the powerset of $\mathbf{V}$, and each directed edge $(V_1, V_2) \in E$ (alternatively, $V_1 \rightarrow V_2$) indicates the existence of an improving (monotonicity or importance) flip from V_1 to V_2, i.e., $V_2 \succ V_1$.*

We now define two terms and one lemma that will be useful in later proofs.

Definition 5 *Let $G(\boldsymbol{C}) = (N, E)$ be the preference graph of a CI-net $\boldsymbol{C}$.*

- *An edge in E is a* monotonicity edge *if and only if it corresponds to a monotonicity flip in $\boldsymbol{C}$.*
- *An edge in E is an* importance edge *if and only if it corresponds to an importance flip in $\boldsymbol{C}$.*

Note that an edge in a CI-net's preference graph may be both a monotonicity edge and an importance edge. As an example, consider the importance graph of a CI-net $\mathbf{C}$ defined over a set of variables $\mathbf{V}$, and let V_1 and V_2 be subsets of $\mathbf{V}$. Suppose the graph contains an importance edge (V_1, V_2), meaning V_2 is preferred to V_1 according to some CI-statement. If (and only if) $V_1 \subset V_2$, this importance edge (V_1, V_2) is also a monotonicity edge.

It is also possible for an importance graph to have monotonicity and importance edges that directly oppose each other. Given the importance edge (V_1, V_2) from the previous paragraph, there must exist a separate monotonicity edge in the opposite direction (V_2, V_1) if (and only if) $V_1 \supset V_2$. This edge forms a cycle with the importance edge (V_1, V_2).

2.2 Satisfiability

A CI-net is *satisfiable* if and only if it does not possess any cycle of flips (Proposition 2 in [Bouveret *et al.*, 2009]). Because each flip in the CI-net's semantics gives rise to exactly one corresponding edge in the CI-net's preference graph, a CI-net is also satisfiable if its preference graph is acyclic (Proposition 4 in [Bouveret *et al.*, 2009]). From this satisfiability condition, Santhanam *et al.* [2010] developed an approach that uses model checking tools to verify satisfiability and perform dominance testing for a CI-net. This approach requires the preference graph to be constructed explicitly and encoded into the input language of the model checker before satisfiability checking begins. The model checker then decides whether the preference graph is acyclic.

Although the problem of deciding the satisfiability of a CI-net is PSPACE-complete [Bouveret *et al.*, 2009], the preference graph of a CI-net that is defined over N variables can be checked for cycles in $O(2^N)$ time, so this problem is feasible for CI-nets that are defined over relatively few variables. Unfortunately, as we will see in Section 4, the model checking-based approach is unacceptably slow for larger, more complex CI-nets because of the additional $\Theta(2^N)$ time required to build and encode the preference graph. Any other technique for checking a CI-net's satisfiability by directly analyzing its preference graph suffers from a similar bottleneck: though the graph analysis itself may be efficient (with a time cost that is linear in the size of the graph), the cost to construct an encoding of the graph for automated analysis dominates the cost to analyze the encoded graph. This bottleneck must be avoided or reduced in order to improve the practical applicability of CI-nets for automating preference reasoning over large, complex sets of qualitative preferences.

3 Verifying CI-Net Satisfiability Without a Preference Graph

We now introduce a different approach for verifying a CI-net's satisfiability, which avoids the performance bottleneck imposed by constructing the full preference graph for the CI-net. Our approach uses properties of the CI-net semantics to efficiently generate the set of all importance flips specified by the CI-net's statements, then uses this set of importance flips to simulate analysis of the CI-net's full preference graph.

We begin by proving several results that lead to a necessary and sufficient condition for satisfiability of a CI-net. This condition is equivalent to the condition defined by [Bouveret *et al.*, 2009], but it can be verified for a given CI-net using only the CI-net's importance flips; the full preference graph is not needed. Once this satisfiability condition is defined, we describe an efficient algorithm for verifying this condition.

3.1 Implicitly Modeling Monotonicity Flips

An advantage of checking the satisfiability of a CI-net using a preference graph is that every improving (or worsening) flip induced by a CI-net is represented explicitly within its preference graph. This allows us to apply existing algorithms for identifying cycles in a directed graph, with correctness guaranteed by the process used to construct the graph. However, there is no need to explicitly represent the monotonicity edges in a CI-net's preference graph, since monotonicity flips are defined with respect to the subset relation over sets of variables. In fact:

Theorem 1 *Any two CI-nets $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ defined over the same number of preference variables have the same set of monotonicity flips (edges), down to renaming of variables.*

Proof. From the definition of a monotonicity flip. □

Furthermore, it is not necessary to check every monotonicity flip to see whether it is part of a cycle. The following theorems show why this is the case.

Definition 6 *An* empty CI-net *is a CI-net that contains no CI-statements.*

Theorem 2 *Every empty CI-net is satisfiable.*

Proof. Suppose in contradiction that an empty CI-net $\mathbf{C}$ over a set of preference variables $\mathbf{V}$ is not satisfiable. Then $\mathbf{C}$ possesses a cycle of flips $V_1 \succ \cdots \succ V_k \succ V_1$, where $V_1, \ldots, V_k$ are unique subsets of $\mathbf{V}$ (i.e., distinct elements of $2^{\mathbf{V}}$). Because $\mathbf{C}$ is empty, it contains no CI-statements, so all flips of $\mathbf{C}$ are monotonicity flips (by Definition 5).

Suppose the cycle contains one flip from V_1 to itself. (Then the importance graph of $\mathbf{C}$ contains a self-loop $V_1 \rightarrow V_1$.) Since all flips of $\mathbf{C}$ are monotonicity flips, $V_1 \subset V_1$, which is not possible.

Suppose the cycle is a sequence of k flips $V_1 \succ \cdots \succ V_k \succ V_1$, where $k > 1$. Since all flips of $\mathbf{C}$ are monotonicity flips, the flip $V_j \succ V_i$ exists if and only if $V_i \subset V_j$. Therefore, $V_1 \supset \cdots \supset V_k \supset V_1$, which is not possible. □

Theorem 3 *Every cycle of two or more flips of a CI-net includes at least one importance flip.*

Proof. A CI-net may have no importance flips, exactly one importance flip, or more than one importance flip. This result holds trivially for CI-nets with no importance flips (i.e., empty CI-nets), since they have no cycles (by Theorem 2).

Consider the set of all CI-nets that have exactly one importance flip. For all positive integers $n > 0$, every CI-net with n variables induces exactly the same set of monotonicity flips as every other CI-net with n variables (Theorem 1). The only possible difference between the semantics of two CI-nets with n variables is in their set of importance flips.

A CI-net with one variable ($n = 1$) induces exactly one monotonicity flip. We do not consider one-flip cycles (self-loops), so the cycle's second flip must be our importance flip. Therefore, the theorem holds in this case.

Now choose an integer $n > 1$ and choose an arbitrary CI-net $\mathbf{C}$ with n variables, such that $\mathbf{C}$ has exactly one importance flip and at least one cycle of flips. Choose an arbitrary cycle of flips in $\mathbf{C}$. This cycle either does or does not contain the only importance flip of $\mathbf{C}$.

Suppose this cycle does not contain the importance flip. Since every flip is a monotonicity flip, an importance flip, or both (Definition 5), all flips in this cycle must be monotonicity flips. As in the proof of Theorem 2, we then have a path of k flips $V_1 \succ \cdots \succ V_k \succ V_1$, where $k > 1$ and each V_i is a unique subset of the CI-net's variables $\mathbf{V}$. Since all flips in the cycle are monotonicity flips, it follows that $V_1 \supset \cdots \supset V_k \supset V_1$, which is not possible. Therefore, our chosen cycle must contain the one importance flip in the preference graph.

This can immediately be generalized to preference graphs that contain more than one importance flip. In such a graph, any cycle that used only monotonicity flips would imply the existence of two sets V_1 and V_k, where $V_1 \supset V_k \supset V_1$. Any cycle must therefore contain one or more importance flips. □

We can use this result to rewrite the necessary and sufficient condition for satisfiability in the following way.

Theorem 4 *A CI-net $\boldsymbol{C}$ is satisfiable if and only if it has no importance flips that participate in a cycle of flips.*

Proof. Immediate from Proposition 2 in [Bouveret *et al.*, 2009] and from Theorem 3. □

3.2 Extracting a CI-Net's Importance Flips

Since checking for a monotonicity flip from one variable set V_i to another variable set V_j is as simple as checking whether $V_i \subset V_j$, it is sufficient for the model of a CI-net's semantics to encode only the importance flips. Because the *ceteris paribus* semantics of CI-net allows each CI-statement to induce multiple importance flips, we must explicitly identify each importance flip that is implied by a CI-statement. Algorithm 1 takes a CI-net $\mathbf{C}$ defined over the set of variables V and uses it to construct the set $\mathbf{C}_{Imp}$ of all importance flips induced by $\mathbf{C}$. Each pair (V_i, V_j) in $\mathbf{C}_{Imp}$ denotes an importance flip $V_j \succ V_i$ specified by the original CI-net $\mathbf{C}$; in turn, this flip is represented by an importance edge $V_i \rightarrow V_j$ in the CI-net's preference graph. This fact — that each pair (V_i, V_j) in $\mathbf{C}_{Imp}$ represents at once both an importance flip and a preference-graph edge — is essential to the correctness of our approach.

Algorithm 1 Extract all importance flips from a CI-net

```
function ImportanceFlips(C, V)
    C_Imp ← ∅                                   ▷ C_Imp ⊆ 2^V × 2^V
    for all statements S+, S− : S1 ≻ S2 ∈ C do
        for all γ ⊆ V \ (S+ ∪ S− ∪ S1 ∪ S2) do
            C_Imp ← C_Imp ∪ {(γ ∪ S+ ∪ S2, γ ∪ S+ ∪ S1)}
        end for
    end for
    return C_Imp
end function
```

Theorem 5 *Given a CI-net $\mathbf{C}$, the set $\mathbf{C}_{Imp}$ returned by* ImportanceFlips*($\mathbf{C}$) contains the pair (V_i, V_j) if and only if the preference graph $G(\mathbf{C})$ contains the edge $V_i \to V_j$.*

Proof. Immediate from Algorithm 1 and Definition 4. □

3.3 Using Importance Flips to Identify Cycles

To show that a CI-net $\mathbf{C}$ is satisfiable, it is necessary and sufficient to show that $\mathbf{C}$ does not contain a cycle of flips (Proposition 2 in [Bouveret *et al.*, 2009]). We now have an implicit representation of all monotonicity flips via the subset relation, and we can construct an explicit representation of all importance flips using the ImportanceFlips function defined in Algorithm 1. At this point, we have all that we need to reconstruct the CI-net's preference graph and apply a standard cycle detection algorithm to decide whether the preference graph is acyclic, which is sufficient to decide the CI-net's satisfiability (by Proposition 4 in [Bouveret *et al.*, 2009]).

Unfortunately, this is inefficient, since $\Theta(2^{|\mathbf{V}|})$ time is needed to build the preference graph before beginning the cycle-detection algorithm [Oster *et al.*, 2015]. Since we already know that monotonicity edges alone cannot form a cycle in the preference graph, we can focus on checking whether each importance flip is part of a cycle. In this subsection, we prove several more results that allow us to focus our satisfiability-checking efforts on a subset of the importance flips without compromising correctness.

Definition 7 *An importance flip (or edge) of a CI-net $\mathbf{C}$ is*

- widening *if* $|V_1| > |V_2|$,
- steady *if* $|V_1| = |V_2|$, *or*
- narrowing *if* $|V_1| < |V_2|$.

These classes form a partition of the set of importance flips (or edges) of $\mathbf{C}$.

Observe that if the CI-net $\mathbf{C}$ is defined using improving-flip semantics (as described in Section 2), widening flips move in the same direction as monotonicity flips, while narrowing flips move in the opposite direction. If worsening-flip semantics are used, the monotonicity preference will be reversed; in this case, the uses of widening and narrowing flips in the following proofs must be interchanged.

Theorem 6 *Every CI-net with exactly one widening importance flip and no other importance flips is satisfiable.*

Proof. Choose such a CI-net $\mathbf{C}$. Let $V_1 \succ V_2$ be the only importance flip of $\mathbf{C}$; since it is a widening flip, $|V_1| > |V_2|$ (by Definition 7). By Theorem 3, if there is a cycle of flips of $\mathbf{C}$, then $V_1 \succ V_2$ must participate in it; if this is the case, then V_2 must be reachable from V_1.

Since $|V_1| > |V_2|$, there is no monotonicity flip from V_1 to V_2, and there is no monotonicity flip from V_1 to any subset of V_2 (from which V_2 could be reached by another monotonicity flip). But there is also no importance flip from V_1 to V_2 or any of V_2's subsets, because the CI-net has no narrowing importance flips. As a result, V_2 is not reachable from V_1, so $\mathbf{C}$ has no cycles and is therefore satisfiable. □

This result generalizes immediately to CI-nets with more than one widening importance flip.

Theorem 7 *Every CI-net with zero or more widening importance flips, but no other (steady or narrowing) importance flips, is satisfiable.*

Proof Sketch. This can be proven by induction on the number n of widening importance flips of a given CI-net $\mathbf{C}$. When $n = 0$, we have an empty CI-net, so the theorem holds by Theorem 2; when $n = 1$, the theorem holds by Theorem 6; and when $n > 1$, the theorem holds by an argument similar to that in the proof of Theorem 6. □

We can use these results to rewrite the satisfiability condition once more.

Theorem 8 *A CI-net $\mathbf{C}$ is satisfiable if and only if it has no steady or narrowing importance flips that are part of a cycle.*

Proof. Immediate from Proposition 2 in [Bouveret *et al.*, 2009] and from Theorem 7. □

3.4 Algorithm

The satisfiability condition in Theorem 8 immediately implies an effective algorithm for verifying the satisfiability of a CI-net: decide whether the CI-net induces any steady or narrowing importance flips that participate in a cycle of flips. We can express the algorithm as a composition of three functions: IsSatisfiable, InCycle, and ImportanceFlips. These functions are defined in Algorithm 2, except that the ImportanceFlips function was defined in Algorithm 1.

The InCycle function returns true if there exists a sequence of steps that can be used to rewrite a given importance flip $V_j \succ V_i$ as an invalid flip $V_i \succ V_i$. Each step in such a sequence involves choosing an improving (or worsening) flip that allows a new set of CI-variables to be substituted into the left side of the statement. In doing so, we traverse the graph implicitly, without the need for an explicit graph construction. The InCycle function performs this traversal, checking at each recursive call whether (1) the cycle beginning and ending at V_i can be completed with zero or more monotonicity flips from V_k or (2) any outgoing preference edges from V_k to some other variable set V_ℓ can be used to rewrite the left side of the preference statement. If neither (1) nor (2) is true, InCycle returns false; if (1) is true, InCycle returns true; and if (1) is false but (2) is true, InCycle tries all possible rewritings of the left side of the preference statement to see if any of them can be used to complete a cycle.

Algorithm 2 Decide if CI-net is satisfiable

```
function IsSatisfiable(C)
    let C_Imp = ImportanceFlips(C)
    for all V_j ≻ V_i ∈ C_Imp where |V_j| ≤ |V_i| do
        if InCycle(C_Imp, V_j ≻ V_i, {}, |V_i|) then
            return false
        end if
    end for
    return true                          ▷ if no statement is in a cycle
end function

function InCycle(C_Imp, V_k ≻ V_j, Path, n)
    for all V_m ≻ V_ℓ ∈ Path do
        if V_k ⊆ V_ℓ then    ▷ monotonicity completes a cycle
            return true
        end if
    end for
    Path = Path ∪ {V_k ≻ V_j}                ▷ append flip to path

        ▷ Is any importance flip from V_k part of a cycle?
    for all V_ℓ ≻ V_k ∈ C_Imp do
        if InCycle(C_Imp, V_ℓ ≻ V_k, Path, n) then
            return true
        end if
    end for

        ▷ Is any monotonicity flip from V_k to a variable-set
        ▷ of size between (|V_k| + 1) and n part of a cycle?
    for all V_ℓ ⊆ V where V_k ⊂ V_ℓ and |V_ℓ| ≤ n do
        if InCycle(C_Imp, V_ℓ ≻ V_k, Path, n) then
            return true
        end if
    end for
    Path = Path \ {V_k ≻ V_j}                ▷ remove flip from path
    return false
end function
```

Theorem 9 *A given CI-net* $\mathbf{C}$ *is satisfiable if and only if the* IsSatisfiable *function in Algorithm 2 returns true when invoked with* $\mathbf{C}$ *as its argument.*

Proof Sketch. The ImportanceFlips function in Algorithm 1 returns $\mathbf{C}_{Imp}$, which is the set of all importance flips induced by the semantics of $\mathbf{C}$. By Theorem 5, $\mathbf{C}_{Imp}$ is equivalent to the set of all importance edges in the preference graph of $\mathbf{C}$. For any given importance flip $V_j \succ V_i$, the InCycle function returns true exactly when a sequence of improving importance flips is constructed (by recursion) from V_i to V_i via V_j. Because IsSatisfiable invokes the InCycle function on every steady or narrowing importance flip of $\mathbf{C}$ and returns true if any one call to InCycle returns true, the satisfiability condition of Theorem 8 is fulfilled by this algorithm. □

4 Empirical Evaluation

We evaluated our satisfiability-checking method by running Java implementations of both our method and the model checking-based method of Santhanam *et al.* [2010] on 687 randomly generated CI-nets, then comparing the time and memory used by each approach. The CI-nets in the test set were defined over 5 to 16 binary variables, with either 5 or 10 CI-statements. The test set contained 30 CI-nets with each combination of n variables and m statements, except fewer CI-nets with 16 variables were tested (15 CI-nets with 5 statements and 12 CI-nets with 10 statements).

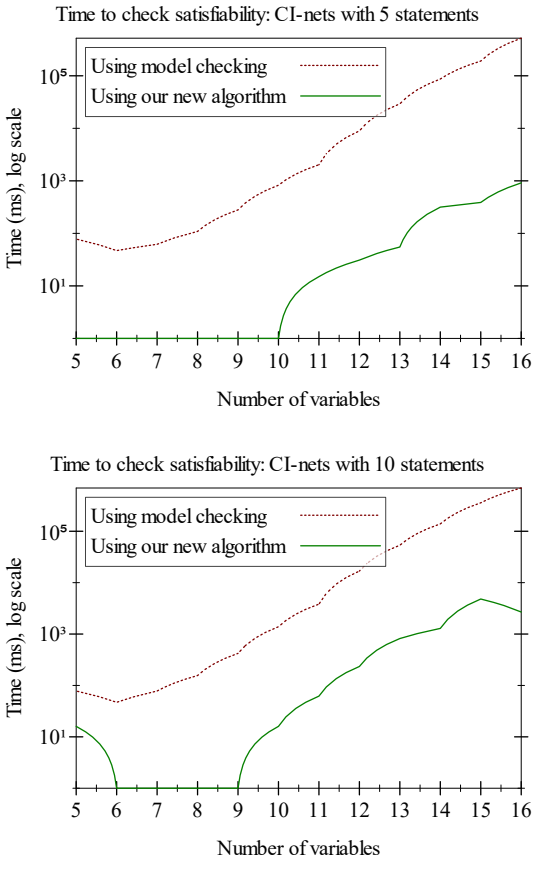

Figure 1: Median time needed to check satisfiability of CI-nets with (a) 5 statements and (b) 10 statements.

Although verifying a CI-net's satisfiability is PSPACE-complete in general, our empirical evaluation shows that our algorithm tends to significantly reduce the average time and memory needed to check the consistency of many CI-nets, compared to the model checking-based approach that constructs the entire preference graph. Figures 1 and 2 show the median time and memory that both approaches used to check the consistency of a CI-net defined over varying numbers of variables, with either 5 or 10 CI-statements. In most cases, our algorithm reduces time usage by about two orders of magnitude compared to the model checking-based method.

However, our method performs poorly on CI-nets that contain statements with a large proportion of narrowing importance flips, sometimes taking several times longer than the model checking-based method. (Our use of medians, rather than means, hides this fact to emphasize the significant improvement in most cases.) This occurs because the InCycle function generates and checks a potentially large set of

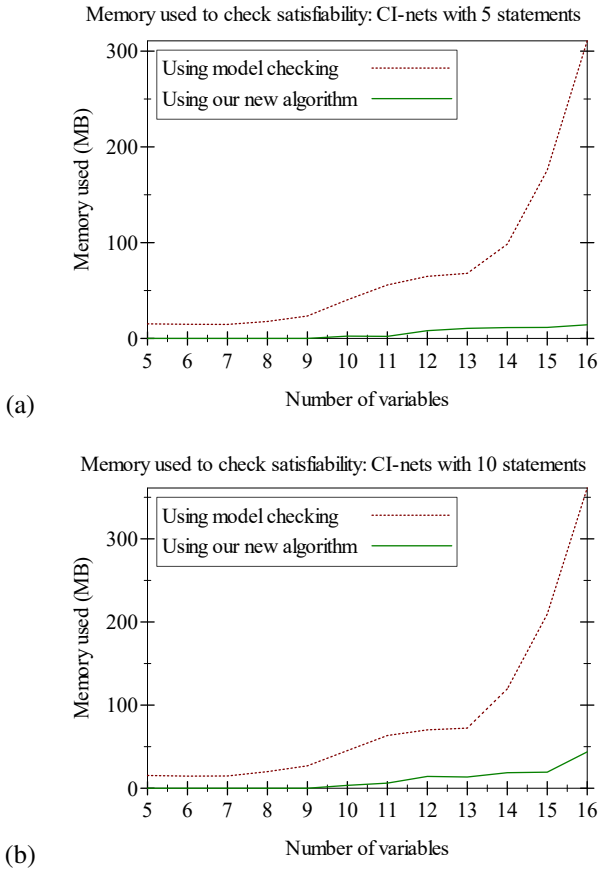

Figure 2: Median memory used to check satisfiability of CI-nets with (a) 5 statements and (b) 10 statements.

monotonicity edges for each narrowing importance flip. The effect is worsened by the fact that our implementation of Algorithm 2 does not perform any memoization, nor does it "prune" improving flips from $\mathbf{C}_{Imp}$ after they are verified.

We are continuing to explore how Algorithm 2 can be modified to improve its efficiency while preserving its correctness. One possibility might be a hybrid approach that uses our method to construct reduced models of parts of the CI-net semantics, which are then verified using model checking.

5 Conclusion and Future Work

We have presented a new algorithm for verifying a CI-net's satisfiability without constructing and analyzing the entire preference graph for the CI-net. Our algorithm is correct because it simulates the behavior of an algorithm that analyzes the entire preference graph, but our algorithm avoids the overhead required to actually construct and analyze the entire graph. In most cases, our algorithm can significantly reduce the time and memory required for satisfiability checking compared to existing methods, making it feasible to automate preference analysis over CI-nets with more variables than existing methods can support.

This algorithm is a step toward our long-term goal of developing a "preference workbench", which will allow users with no special training or experience in qualitative preference reasoning techniques to make better-informed decisions using the expressive power of CI-nets. Such a tool may be especially useful in the area of requirements engineering, where it is common for a wide variety of different stakeholders involved in the design of a software system to express set-based conditional qualitative preferences regarding possible features or properties that they need or desire in the system [Oster *et al.*, 2015]. When completed, our "preference workbench" will include facilities for editing, visualizing, comprehending, and tracing preferences within a CI-net (or related) preference model. We look forward to seeing how all of these features will be enabled by our efficient algorithm for verifying the satisfiability of a CI-net.

References

[Bouveret *et al.*, 2009] Sylvain Bouveret, Ulle Endriss, and Jérôme Lang. Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st Intl. Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 67–72, 2009.

[Oster *et al.*, 2012] Zachary J. Oster, Ganesh Ram Santhanam, Samik Basu, and Vasant Honavar. Model checking of qualitative sensitivity preferences to minimize credential disclosure. In Corina Pasareanu and Gwen Salaün, editors, *FACS*, volume 7684 of *Lecture Notes in Computer Science*, pages 205–219. Springer, 2012.

[Oster *et al.*, 2015] Zachary J. Oster, Ganesh Ram Santhanam, and Samik Basu. Scalable modeling and analysis of requirements preferences: A qualitative approach using CI-nets. In Didar Zowghi, Vincenzo Gervasi, and Daniel Amyot, editors, *23rd IEEE Intl. Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015*, pages 214–219. IEEE Computer Society, 2015.

[Santhanam *et al.*, 2010] Ganesh Ram Santhanam, Samik Basu, and Vasant Honavar. Dominance testing via model checking. In Maria Fox and David Poole, editors, *Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press, 2010.

[Santhanam *et al.*, 2013a] Ganesh Ram Santhanam, Samik Basu, and Vasant Honavar. Verifying preferential equivalence and subsumption via model checking. In Patrice Perny, Marc Pirlot, and Alexis Tsoukiàs, editors, *Algorithmic Decision Theory - Third International Conference, ADT 2013, Bruxelles, Belgium, November 12-14, 2013, Proceedings*, volume 8176 of *Lecture Notes in Computer Science*, pages 324–335. Springer, 2013.

[Santhanam *et al.*, 2013b] Ganesh Ram Santhanam, Zachary J. Oster, and Samik Basu. Identifying a preferred countermeasure strategy for attack graphs. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, CSIIRW '13, pages 11:1–11:4, New York, NY, USA, 2013. ACM.