# Learning CP-net Preferences Online from User Queries

Joshua T. Guerin[1], Thomas E. Allen[2], and Judy Goldsmith[2]

[1] The University of Tennessee at Martin jguerin@utm.edu
[2] University of Kentucky {teal223,goldsmit}@cs.uky.edu

**Abstract.** We present an online, heuristic algorithm for learning Conditional Preference networks (CP-nets) from user queries. This is the first efficient and resolute CP-net learning algorithm: if a preference order can be represented as a CP-net, our algorithm learns a CP-net in time $n^p$, where $p$ is a bound on the number of parents a node may have. The learned CP-net is guaranteed to be consistent with the original CP-net on all queries from the learning process. We tested the algorithm on randomly generated CP-nets; the learned CP-nets agree with the originals on a high percent of non-training preference comparisons.

## 1 Introduction

To support decision making, an intelligent agent often requires some way to *learn* what a human user prefers and *concisely represent* those preferences. CP-nets [1] offer a potentially compact qualitative representation of human preferences that operate under *ceteris paribus* ("with all else being equal") semantics. In this paper we present a novel algorithm through which an agent learns the preferences of a user. CP-nets are used to represent such preferences and are learned online through a series of queries generated by the algorithm. Our algorithm builds a CP-net for the user by creating nodes and initializing Conditional Preference Tables (CPTs), then gradually adding edges and forming more complex CPTs consistent with responses to queries until a confidence parameter is reached or no further progress can occur. The algorithm does not always converge to the original CP-net, but our experiments show that it can learn a CP-net that closely tracks with the original for a set of outcome comparison queries not used in the learning phase. While one could treat the model learning process as one of replicating the structure of the original (latent) CP-net, we assume here that this is unnecessary, as long as the *preferences modeled by* the two networks differ by very little.

The problem of learning CP-nets from example data is known to be hard in general, even for acyclic, binary CP-nets (the class we consider here), and also for *separable* CP-nets (such as we use in the first phase of our algorithm; see Def. 1) [2, 3]. This has led to a diversity of proposed methods for learning CP-nets: a regression-based approach [4, 5], Angluin-style query learning [6, 7], learning via reduction to 2-SAT [8], and learning a CP-net indirectly via the

exponentially larger preference graph [9]. Our work builds upon this body of previous research, particularly that of [2, 3] and [8], but differs in several notable respects. First, our method is designed for an *active, online* environment in which the agent directly interacts with the user, rather than one in which the user's actions can be observed over time. Second, in our queries we allow for arbitrary outcome comparisons rather than (as in [6, 7]) restricting to *swap* comparisons where outcomes differ in at most one variable. We believe our approach better reflects the rich environment in which human decisions are often made. Third, our algorithm is *robust*; unlike [8], for example, it will always output a CP-net. Finally, given a constant bound on the number of parents, our algorithm can learn a CP-net in polynomial time. This differs in particular from that of [9], which in the worst case is of double-exponential complexity in the number of variables.

The remainder of our paper is organized as follows: In Sec. 2 we review the use of CP-nets to model preferences. In Sec. 3 we present the algorithm itself, followed by analysis in Sec. 4. Section 5 consists of a series of experiments and significant results. We conclude with opportunities for future research.

## 2 Modeling Preferences with CP-nets

Consider a recommender system that assists customers in purchasing a guitar. The customer surely cannot consider every possible guitar, but will buy one that is satisfactory, given her preferences. Various factors differentiate the possibilities, such as BRAND, BODY, and COLOR. We assume the customer can consistently rank alternatives: presented two guitars, she can say, "I prefer the first to the second." We further assume the customer can make more general comparisons, such as, "In general, I prefer the *Fender* BRAND to *Gibson*." But offered several alternatives, that customer may ultimately prefer a *specific* Gibson guitar based on other factors; that is, the user's preferences may be *conditional*.

More formally, by *preference*, we mean a strict partial order $\succ$ over a finite set of *outcomes* $\mathcal{O}$ by a user. Such outcomes can be factored into *variables* $\mathcal{V}$ with associated (binary) *domains* $\mathrm{Dom}(\mathcal{V})$: $\mathcal{O} = v_1 \times v_2 \times \cdots \times v_n$. We call $o[i]$ the *projection* of outcome $o$ onto variable $v_i$ and $o[U]$ the projection onto a *set* of variables $U \subseteq \mathcal{V}$. Note that the number of outcomes and orderings is exponential in the number of variables. CP-nets can offer a more compact representation.

**Definition 1.** *A* CP-net $\mathcal{N}$ *is a directed graph. Each node* $v_i$ *represents a preference over a finite domain. An edge* $(v_i, v_j)$ *indicates that the preference over* $v_j$ *depends on the value of* $v_i$*. If a node has no incoming edges, the preference involving its variable is not conditioned on values represented elsewhere in the graph. A* separable CP-net *is one in which no variable depends on any other.*

**Definition 2.** *A* conditional preference table (CPT) *is associated with each node* $v_i$ *and specifies the preference over* $\mathrm{Dom}(v_i)$ *as a function of the values assigned to its* parent nodes $\mathrm{Pa}(v_i)$*. If a CPT has an entry for every combination of values from the domains of its parents, we say it is* complete*.*

We define the *size* of a CPT as its number of rows. The size of a CP-net is the sum of the sizes of its CPTs. One can observe that if a node $v_j$ has incoming edges from $k$ parents, each representing binary variables, then size(CPT($v_j$)) = $2^k$. Thus size grows exponentially in the number of parents. To guarantee tractability, we make some simplifying assumptions: 1. Cycles are disallowed. 2. We restrict to binary domains. 3. A maximum bound $p$ is placed on the number of parents a node may have: We conjecture that most human preferences are conditioned on 3–5 nodes and thus feel justified in assuming such a bound.

**Example 1.** *Consider the CP-net to the right representing a conditional preference: The edge from* BRAND *to* COLOR *indicates that the customer's preference of* COLOR *depends on* BRAND. *The CPTs associated with each node provide the ordering. In general, the customer prefers* Fender *to* Gibson. *If a* Fender *is available, she prefers* red, *but if only a* Gibson *is available, she prefers* gold.

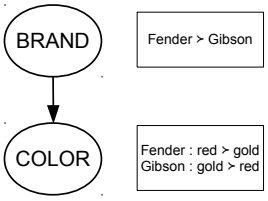


**Fig. 1.** Simple CP-net

## 3 Algorithm

Our algorithm consists of two phases. First, it constructs a separable CP-net with default CPTs. Next, it successively attempts to refine the model, adding edges and learning more complex CPTs consistent with evidence from *user queries.* (See LEARN-CP-NET and its subroutine FIND-PARENTS [Alg. 1 and 2]).

### 3.1 Phase 1: The Separable CP-net Basis

In Phase 1 (Alg. 1, lines 1–9), our algorithm constructs a *separable CP-net basis* by asking the user to provide a default preference for each $v_i \in \mathcal{V}$. This initial CP-net could be characterized as a first impression of the user's preferences.

**Definition 3.** *Let $v_i$ be a binary variable with* $\mathrm{Dom}(v_i) = \{x_i, y_i\}$. *An* attribute comparison query *is one in which we ask the user whether $x_i \succ y_i$ or $y_i \succ x_i$.*

Here we assume that the user is able to reflect on possible outcomes and discern what she prefers most of the time. (In our experiments, as discussed in Sec. 5.1, we *simulate* the user's response to such queries by reporting a preference that occurs most frequently in CPT($v_i$).) The result is a CP-net with default CPTs and no edges. If we were *confident* that the user's preference over each attribute did not depend on the value of any other attribute, we could return $\mathcal{N}$ as *the* CP-net that consistently modeled the user's preferences. At this point, however, we are *unconfident* of each node's parentage. We maintain disjoint sets *Confident* and *Unconfident* such that $v_i \in$ *Confident* iff enough data has been collected via user queries to conclude that the preferences over $v_i$ are conditioned only by its parent variables in the graph of $\mathcal{N}$; otherwise $v_i \in$ *Unconfident*. Initially *Unconfident* = $\mathcal{V}$ and *Confident* = $\emptyset$.

**Example 2.** *Nodes* BODY *and* BRAND *have been inserted into* $\mathcal{N}$ *along with their default CPTs. A node and CPT must now be created for* COLOR. *The agent asks the customer, "In general, do you prefer a* ***guitar*** *that is* red *or* gold*?" The user replies that* gold *is usually preferred, and a node and CPT is inserted. The resulting CP-net basis is shown.*


**Fig. 2.** Separable CP-net Basis

**Algorithm 1** LEARN-CP-NET($\mathcal{V}$, $p$, $q$)

**Input:** $\mathcal{V}$ a set of binary variables
$p$ maximum number of parents
$q$ confidence threshold parameter

**Global:** *Comparisons* responses to user queries
*Confident* set of learned nodes

**Output:** $\mathcal{N}$ the CP-net learned from the user

$\mathcal{N} \leftarrow \emptyset$
*Comparisons* $\leftarrow \emptyset$
*Confident* $\leftarrow \emptyset$
*Unconfident* $\leftarrow \mathcal{V}$
**for** $v_i \in \mathcal{V}$ **do**
  query user: do you prefer $x_i \succ y_i$ or $y_i \succ x_i$?
  $v_i$.CPT $\leftarrow$ default CPT based on user response
  insert $v_i$ into $\mathcal{N}$
**end for**
**repeat**
  **for** $r \leftarrow 0$ **to** $p$ **do**
    **for** $v_i \in$ *Unconfident* **do**
      $(P, C) \leftarrow$ FIND-PARENTS($v_i, r, q$)
      **if** $C \neq$ FAIL **then**
        $v_i$.CPT $\leftarrow C$
        add edges from all $P$ to $v_i$
        move $v_i$ from *Unconfident* to *Confident*
      **end if**
    **end for**
  **end for**
**until** no parents added this iteration
**return** $\mathcal{N}$

### 3.2 Phase 2: Refining the CP-net Model

In Phase 2 we refine $\mathcal{N}$ by discovering such conditional relationships as may exist between variables by asking the user's preference over pairs of outcomes.

**Algorithm 2** FIND-PARENTS$(v_i, r, q)$

**Input:** $v_i$ node representing $i$th variable
$r$ the number of parents in the trials
$q$ confidence threshold parameter

**Global:** *Comparisons* responses to user queries
*Confident* set of learned nodes

**Output:** $C$ a CPT with $2^r$ rows or FAIL
$P$ newly parents discovered for $v_i$ or $\emptyset$

```
1: for P ∈ {all subsets of Confident of size r} do
2:    (C, evidCount) ← CREATE-CPT(v_i, P)
3:    while (C ≠ FAIL) and (evidCount < q) do
4:       (o_1, o_2) ← generate random query for v_i
5:       query user: do you prefer outcome o_1 or o_2?
6:       add o_1, o_2 to Comparisons in specified order
7:       (C, evidCount) ← CREATE-CPT(v_i, P)
8:    end while
9:    if C ≠ FAIL then
10:      return (P, C)
11:   end if
12: end for
13: return (∅, FAIL)
```

**Definition 4.** *In an* outcome comparison query*, we provide the user a pair of outcomes,* $\{o_1, o_2\} \in \mathcal{O}$ *such that* $o_1 \neq o_2$*. The user responds with* $o_1 \succ o_2$*,* $o_2 \succ o_1$ *or* $o_1 \bowtie o_2$*, respectively indicating that she strictly prefers the first outcome to the second, the second to the first, or is unable to state a preference.*

If the user is able to answer an outcome comparison query with either $o_1 \succ o_2$ or $o_2 \succ o_1$, we treat the response as *evidence* of the user's underlying preference model. If the user cannot state a preference, we treat that as an indication that the two outcomes are *incomparable*. We expect that the user will provide consistent answers to the same outcome query and hence ensure that each unordered pair of outcomes is asked at most once. Responses are stored in a *Comparisons* database, gradually adding to the evidence used to construct the model CP-net. Outcome pairs are generated *randomly*, with the constraint that the pair must be *relevant* to the node under consideration and not already in *Comparisons*.

**Definition 5.** *For any given* $v_i$*,* random query *is an outcome comparison query in which* $o_1$ *and* $o_2$ *are selected uniformly randomly from their domains, with the requirement that the query must be* relevant *to node* $v_i$*:* $o_1[i] \neq o_2[i]$*. A* random adaptive query *adds the additional requirement that for all* $v_j \in$ Confident*,* $o_1[j] = o_2[j]$*.*

*Random adaptive* queries provide a heuristic that may reduce the search space for a CP-net by not continuing to analyze nodes once they are labeled *Confident*.

Using random queries, our search proceeds as follows. In the `repeat–until` loop of LEARN-CP-NET (Alg. 1, lines 10–21), we search first for nodes that do not need parents (i.e., nodes that represent features for which the user's preferences are unconditional). For each *target node* $v_i \in \mathcal{V}$, we call FIND-PARENTS. If FIND-PARENTS is confident that the preferences over $v_i$ are unconditional, it returns $(\emptyset, C)$, where $C$ is a CPT for $v_i$ consistent with all queries stored in *Comparisons*; otherwise, it returns ($\emptyset$, FAIL), indicating failure. In the former case, the default CPT of $v_i$ is replaced with $C$ and $v_i$ is reclassified as *Confident*; in the latter, the search continues with the next *Unconfident* node.

As long as *Unconfident* $\neq \emptyset$, we continue trying to refine our model with new conditional relationships, represented as edges and correspondingly more complex CPTs. For each remaining *Unconfident* node, we iterate over potential sets of parent nodes, starting with single-parent relationships, then two parents, three, and so on, up to the bound on parents $p$. If a set $P$ of parents can be found, edges are added from each newly discovered parent to target node $v_i$, the default CPT is replaced with $C$, and $v_i$ is reclassified as *Confident*. If at any point, however, we iterate from 0 to $p$ and fail to add parents to *any* node, we stop refinement and are satisfied with the CP-net that we have thus generated to that point, even if some nodes remain *Unconfident*. For such nodes, the default CPT assigned in Phase 1 would be output in the finalized CP-net. Indeed, in the worst case, it is possible that *all* of the nodes could be output with the CPTs in their default state. However, in practice we find that this rarely happens. In all cases, though, LEARN-CP-NET will return a CP-net; it will never output failure.

**Example 3.** *Algorithm* LEARN-CP-NET *has determined from outcome comparison queries that* BODY *and* BRAND *each has 0 parents. As such, their status was moved from* Unconfident *to* Confident*, and the entries in their CPTs are thus finalized. However, the status of* COLOR *at this point is still undetermined.*

*An inspection of* Comparisons *indicates that the preference over* COLOR *is* not *independent. Sometimes the customer prefers* gold*, sometimes* red*. Additional queries are generated; e.g., "Do you prefer a* heavy gold Gibson *or a* heavy red Gibson*?" Nonetheless, all attempts to prove that* BODY *or* BRAND *is the* sole *parent of* COLOR *fail to yield a CPT.*

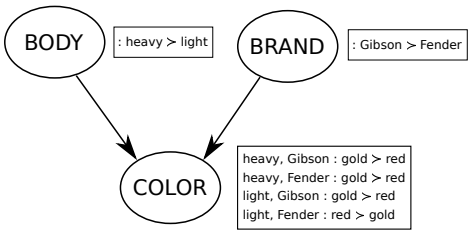


**Fig. 3.** The Fully Learned CP-net

*The algorithm next attempts to establish whether* both BODY and BRAND *are parents of* COLOR*. This time a CPT is produced and sufficient evidence is gathered. While the user does indeed generally prefer* gold*, she prefers a* red ***guitar*** *only if it is* light *and a* Fender*. Figure 3 shows completed CP-net* $\mathcal{N}$*.*

Subroutine FIND-PARENTS iterates over all subsets of *Confident* of the specified size $r$. For each subset $P$, FIND-PARENTS calls CREATE-CPT to determine

whether $P$ is consistent with all queries stored in *Comparisons*. For a given target node and set of possible parents, CREATE-CPT constructs a 2-SAT instance such that (1) a satisfying assignment tells us that the target node's values are consistent with the given set of parents and (2) the assignment to variables gives us the entries of the target node's CPT. Our method for this closely follows [8], to which the reader is referred for specifics. It should be noted that this method converges to the original network only for a restricted class of CP-nets; however, our objective is not to recover the original network, but to learn one that closely approximates the original for possible outcome comparison queries. If the CPT returned from CREATE-CPT is not complete, the default values from Phase 1 are used to provide the ordering. FIND-PARENTS continues generating random queries and calling CREATE-CPT until the *evidence* to support $P$ as the parents of $v_i$ reaches a specified confidence threshold $q$. Specifically, the *evidence count* is the number of queries in *Comparisons relevant* to node $v_i$, and *confidence parameter* $q$ is the number of outcome pairs required before we are confident that $P$ is in fact the parent set of $v_i$. (We discuss the use of the $q$ threshold for minimum evidence in Section 3.2 and metrics for confidence in 5.)

## 4 Analysis

**Theorem 1.** LEARN-CP-NET *is* resolute*—that is, it is guaranteed to output a consistent CP-net* $\mathcal{N}$*.*

*Proof.* (Sketch.) An initial $\mathcal{N}$ is created in the algorithm's phase 1. The repeat–until loop that follows will iterate at most once per variable. Since $\mathcal{V}$ is finite, the algorithm will terminate and output a CP-net. Moreover, since edges are added only from nodes in *Confident* to those in disjoint set *Unconfident*, $\mathcal{N}$ will not contain a cycle; hence it will be consistent.

**Theorem 2.** *The learning algorithm* LEARN-CP-NET *is time polynomial in* $n^p$ *and* $q$ *in the worst case.*

*Proof.* (Sketch.) Using Tarjan's algorithm, we implement the CREATE-CPT 2-SAT algorithm in linear time [10]. Given this, the polynomial time complexity of LEARN-CP-NET and FIND-PARENTS follows directly from Alg. 1 and 2.

## 5 Experiments

### 5.1 Experiment Design

We evaluate the effectiveness of our algorithm through an experimental approach. First, we generate a random CP-net training model $\mathcal{N}_T$ that simulates the preferences of a hypothetical user. We then apply LEARN-CP-NET to the model, posing queries that are answered by an agent on the basis of $\mathcal{N}_T$. Finally, we evaluate the success of the learned model $\mathcal{N}_L$ in terms of the likelihood that it will correctly predict the response of $\mathcal{N}_T$ to a randomly drawn pair of outcomes

from $\mathcal{O}$. We assume here that a user's preferences can be modeled by a CP-net. (Whether human preferences can actually be modeled by CP-nets is a subject the authors are presently exploring through interdisciplinary research.) We further assume that the attribute variables and their binary values are common knowledge and not additional parameters that must be learned.

A number of questions guided our experiments, such as: How many queries are required to learn a CP-net model? Does the choice of outcome comparison querying strategies (random and adaptive) affect CP-net learnability? How does quality of CP-net learning change as the size and density of its graphical structure changes? Do algorithm runtimes conform to expectations?

To evaluate the algorithm, we generated a set of *random CP-nets*. Since variables and domains are common knowledge, the nodes of $\mathcal{N}_T$ are an input to the generation algorithm, as are the number of directed edges $e$ in the graph and the maximum number of parents $p$. (Note that while $\mathcal{V}$ is the same for $\mathcal{N}_T$ and $\mathcal{N}_L$, $e$ and $p$ may differ.) The edges of $\mathcal{N}_T$ are inserted at random with equal probability, such that no node has more than $p$ parents and no cycles are introduced. Complete CPTs for each node are then generated by selecting, again uniformly randomly, an entry $x_i \succ y_i$ or $y_i \succ x_i$ for each row (with the provision that the CPT implies dependence on the value of each parent node).

**Algorithm 3** GENERATE-RANDOM-CP-NET($\mathcal{V}, e, p$)

**Input:** $\mathcal{V}$ a set of binary variables
$e$ number of edges in graph
$p$ maximum number of parents

**Output:** $\mathcal{N}_T$ the randomly generated CP-net

initialize $\mathcal{N}$: nodes $\mathcal{V}$, no edges, empty CPTs
**for** $k \leftarrow 1$ **to** $e$ **do**
  add a randomly selected edge $(v_i, v_j)$ to $\mathcal{N}_T$ s.t. $1 \leq i < j \leq n$ and $|\mathrm{Pa}(v_j)| \leq p$
**end for**
**for** $k = 1$ **to** $|\mathcal{V}|$ **do**
  **for** each of the $2^{|\mathrm{Pa}(v_k)|}$ entries in CPT($v_k$) **do**
    insert $x_k \succ y_k$ or $y_k \succ x_k$ at random
  **end for**
**end for**
**return** $\mathcal{N}_T$

Our simulations do not employ human subjects. The learning algorithm queries an agent that answers on the basis of a given CP-net model $\mathcal{N}_T$. Recall that in Phase 1, an *attribute comparison query* is asked for each binary variable: "In general, do you prefer $x_i$ or $y_i$?" We assume that the human user has the capacity to reflect on such outcomes and determine whether she usually prefers $x_i$ or $y_i$. An agent that simulates such a capacity we call *attribute aware*.

**Definition 6.** *An agent is* attribute aware *if it replies to an attribute comparison query with the preference that occurs in the majority of its CPT entries. If $x_i \succ y_i$ and $y_i \succ x_i$ occur equally often, the agent outputs one of these at random.*

Learn-CP-net also asks the user a series of *outcome comparison queries*: "Do you prefer $o_1 \succ o_2$, $o_2 \succ o_1$, or neither?" In response, the agent consults its CP-net to determine if a preference is entailed. It has been shown that, in general, determining whether one outcome dominates another given a CP-net ($\mathcal{N} \models o_1 \succ o_2$) is hard [11]. However, [1] introduce the weaker notion of an *ordering query*, which we employ in our simulations.

**Definition 7.** *Given outcomes $o_1$ and $o_2$, we say $o_1$ is* consistently orderable *over $o_2$ with respect to $\mathcal{N}$ iff there exists a node $v_i \in \mathcal{N}$ such that* (1) *$o_1$ and $o_2$ assign the same value to all ancestors of $v_i$ and* (2) *$o_1$ assigns a more preferred value to $v_i$ than $o_2$. A search for such a node is called an* ordering query*.*

Rather than asking if $\mathcal{N} \models o_1 \succ o_2$, an ordering query asks if $\mathcal{N} \not\models o_2 \succ o_1$. Note that while $\mathcal{N} \models o_1 \succ o_2$ implies $\mathcal{N} \not\models o_2 \succ o_1$, the reverse does not hold: it may be that $\mathcal{N} \not\models o_1 \succ o_2$ and $\mathcal{N} \not\models o_2 \succ o_1$. As Boutilier, et al. [1] showed, ordering queries can be answered in time polynomial in size($\mathcal{N}$). In response to outcome queries in our simulations, $\mathcal{N}_T$'s agent answers $o_1 \succ o_2$ if $o_1$ is consistently orderable over $o_2$, $o_2 \succ o_1$ if the reverse is true, and $o_1 \bowtie o_2$ if a preference cannot be determined. Note that, according to [1]'s definition of *consistently orderability*, we could have $o_1$ consistently orderable over $o_2$ based on some variables, and vice versa based on others. For purposes of answering ordering queries, we check all variables, and say that $o_1$ is consistently orderable over $o_2$ if there is some variable for which that holds, and *no* variables that witness that $o_2$ is consistently orderable over $o_1$. This is a stronger condition than [1]'s. However, the disadvantage of using ordering queries instead of dominance testing is that an agent using the former is more likely to report that $o_1 \bowtie o_2$, especially as $n$ grows large.

Our primary measure of how well the learned CP-net $\mathcal{N}_L$ agrees with the training model $\mathcal{N}_T$ is to compare directly the preference ordering induced by $\mathcal{N}_L$ with that of $\mathcal{N}_T$ for all unordered pairs of possible outcomes.

**Definition 8.** *Given an outcome comparison involving $o_1$ and $o_2$, we say that CP-net $\mathcal{N}_L$* agrees *with $\mathcal{N}_T$,* disagrees*, or is* indecisive *as follows:*

| $\mathcal{N}_T$ | $\mathcal{N}_L : o_1 \succ o_2$ | $o_2 \succ o_1$ | $o_1 \bowtie o_2$ |
|---|---|---|---|
| $o_1 \succ o_2$ | agrees | disagrees | indecisive |
| $o_2 \succ o_1$ | disagrees | agrees | indecisive |

*The* agreement metric *$M$ is then a vector representing the percentage of total outcome comparisons for which $\mathcal{N}_L$ agrees, disagrees, or is indecisive w.r.t. $\mathcal{N}_T$.*

Note that we do not include in our counts outcome comparisons for which *training* model $\mathcal{N}_T$ is indecisive. Because of this, the agreement metric is not symmetric; that is, in general $M(\mathcal{N}_T, \mathcal{N}_L) \neq M(\mathcal{N}_L, \mathcal{N}_T)$.

Since the number of outcomes is $2^n$ for $n$ binary variables, the number of unordered outcome pairs $\{o_1, o_2\}$ is $O(2^{2n})$. Hence, while our algorithm is of polynomial complexity, our method of evaluation is exponential in the number of variables if we calculate $M$ *exactly*. However, we can *approximate* $M$ satisfactorily through *sampling*. For our sample size we chose $20\,000 > z^2_{\alpha/2}/(4\varepsilon^2)$, where $z_{\alpha/2}$ is obtained from the normal distribution, $\varepsilon$ is the desired bound of $\pm 0.5\%$, and $1 - \alpha$ provides a 95%-confidence interval. We calculate $M$ exactly for $n \leq 7$ and estimate through sampling for $n > 7$.

### 5.2 Experimental Results

Tables 1 and 2 show metrics of $\mathcal{N}_L$ w.r.t. $\mathcal{N}_T$ over a series of experiments for $n$ nodes and confidence threshold $q$. Density $\delta = e/n$ is the desired ratio of edges to nodes ($\delta = \infty$ implies a maximally dense acyclic graph, given the bound $p$ on parents). The results shown are for $p = 5$ and represent averages over 30 trials. The data reflect random adaptive queries and an attribute aware agent; our results for random non-adaptive queries (not shown) reflected slightly lower levels of agreement [12]. Table 2 shows agreement for a higher granularity of $q$, along with the choice of $q$ in this range that maximized agreement ($q*$).

**Table 1.** Agreement of $\mathcal{N}_L$ with $\mathcal{N}_T$

| | Agreement | | | | Disagreement | | | | Indecision | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $q$=5 | 10 | 15 | 20 | $q$=5 | 10 | 15 | 20 | $q$=5 | 10 | 15 | 20 |
| | | | | | $\delta = 1$ | | | | | | | |
| 4 | 0.96 | 0.98 | 0.98 | 0.98 | 0.03 | 0.01 | 0.02 | 0.01 | 0.01 | 0.00 | 0.00 | 0.00 |
| 6 | 0.79 | 0.94 | 0.97 | 0.98 | 0.07 | 0.04 | 0.02 | 0.02 | 0.14 | 0.02 | 0.00 | 0.00 |
| 8 | 0.69 | 0.77 | 0.77 | 0.75 | 0.07 | 0.03 | 0.02 | 0.02 | 0.24 | 0.20 | 0.21 | 0.23 |
| 10 | 0.65 | 0.65 | 0.58 | 0.53 | 0.04 | 0.02 | 0.02 | 0.01 | 0.31 | 0.33 | 0.41 | 0.46 |
| 12 | 0.57 | 0.65 | 0.56 | 0.42 | 0.02 | 0.02 | 0.01 | 0.01 | 0.41 | 0.34 | 0.43 | 0.58 |
| | | | | | $\delta = 2$ | | | | | | | |
| 4 | 0.92 | 0.98 | 0.98 | 0.98 | 0.06 | 0.02 | 0.02 | 0.02 | 0.02 | 0.00 | 0.00 | 0.00 |
| 6 | 0.72 | 0.97 | 0.98 | 0.98 | 0.13 | 0.03 | 0.02 | 0.02 | 0.16 | 0.00 | 0.00 | 0.00 |
| 8 | 0.60 | 0.76 | 0.76 | 0.76 | 0.11 | 0.04 | 0.03 | 0.02 | 0.29 | 0.20 | 0.21 | 0.22 |
| 10 | 0.53 | 0.64 | 0.60 | 0.52 | 0.09 | 0.04 | 0.02 | 0.01 | 0.38 | 0.33 | 0.38 | 0.47 |
| 12 | 0.52 | 0.64 | 0.41 | 0.36 | 0.07 | 0.03 | 0.02 | 0.01 | 0.41 | 0.33 | 0.57 | 0.63 |
| | | | | | $\delta = 3$ | | | | | | | |
| 4 | 0.94 | 0.97 | 0.98 | 0.98 | 0.04 | 0.03 | 0.02 | 0.02 | 0.02 | 0.00 | 0.00 | 0.00 |
| 6 | 0.82 | 0.95 | 0.98 | 0.98 | 0.11 | 0.04 | 0.02 | 0.02 | 0.08 | 0.01 | 0.00 | 0.00 |
| 8 | 0.63 | 0.80 | 0.73 | 0.76 | 0.13 | 0.04 | 0.02 | 0.01 | 0.24 | 0.16 | 0.25 | 0.22 |
| 10 | 0.61 | 0.65 | 0.48 | 0.47 | 0.13 | 0.04 | 0.01 | 0.01 | 0.26 | 0.31 | 0.51 | 0.52 |
| 12 | 0.57 | 0.62 | 0.44 | 0.28 | 0.13 | 0.04 | 0.02 | 0.01 | 0.30 | 0.34 | 0.55 | 0.72 |
| | | | | | $\delta = \infty$ | | | | | | | |
| 4 | 0.91 | 0.98 | 0.98 | 0.98 | 0.06 | 0.02 | 0.02 | 0.02 | 0.03 | 0.00 | 0.00 | 0.00 |
| 6 | 0.84 | 0.96 | 0.97 | 0.98 | 0.11 | 0.03 | 0.03 | 0.02 | 0.05 | 0.01 | 0.00 | 0.00 |
| 8 | 0.66 | 0.76 | 0.76 | 0.75 | 0.13 | 0.03 | 0.02 | 0.01 | 0.21 | 0.21 | 0.22 | 0.24 |
| 10 | 0.62 | 0.56 | 0.48 | 0.40 | 0.13 | 0.04 | 0.02 | 0.01 | 0.25 | 0.39 | 0.50 | 0.59 |
| 12 | 0.54 | 0.54 | 0.37 | 0.26 | 0.13 | 0.04 | 0.01 | 0.01 | 0.33 | 0.42 | 0.61 | 0.74 |

**Table 2.** Agreement for various choices of $q$

| Agreement $\delta = 1$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $q$=6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | $q*$ | max |
| 6 | 0.90 | 0.90 | 0.91 | 0.96 | 0.96 | 0.97 | 0.97 | 0.97 | 0.97 | 13 | 97% |
| 7 | 0.77 | 0.80 | 0.85 | 0.87 | 0.83 | 0.87 | 0.84 | 0.89 | 0.89 | 14 | 89% |
| 8 | 0.69 | 0.75 | 0.77 | 0.83 | 0.77 | 0.84 | 0.77 | 0.80 | 0.79 | 11 | 85% |
| 9 | 0.76 | 0.76 | 0.75 | 0.70 | 0.81 | 0.74 | 0.78 | 0.70 | 0.72 | 10 | 81% |
| 10 | 0.61 | 0.69 | 0.70 | 0.72 | 0.66 | 0.70 | 0.69 | 0.68 | 0.60 | 9 | 72% |

**Table 3.** Effect of decreasing $p$ for $\mathcal{N}_L$ below that of $\mathcal{N}_T$

| | Agreement | | | | | Disagreement | | | | | Indecision | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $p$=1 | 2 | 3 | 4 | 5 | $p$=1 | 2 | 3 | 4 | 5 | $p$=1 | 2 | 3 | 4 | 5 |
| $\mathcal{N}_T$: $p = 5$, $\delta = \infty$ | | | | | | | | | | | | | | | |
| 6 | 0.33 | 0.42 | 0.67 | 0.86 | 0.96 | 0.05 | 0.03 | 0.03 | 0.03 | 0.03 | 0.63 | 0.55 | 0.30 | 0.11 | 0.00 |
| 7 | 0.25 | 0.33 | 0.44 | 0.70 | 0.90 | 0.04 | 0.03 | 0.02 | 0.03 | 0.03 | 0.71 | 0.64 | 0.54 | 0.28 | 0.07 |
| 8 | 0.19 | 0.24 | 0.34 | 0.55 | 0.76 | 0.03 | 0.02 | 0.02 | 0.02 | 0.03 | 0.78 | 0.74 | 0.64 | 0.43 | 0.21 |
| 9 | 0.14 | 0.20 | 0.27 | 0.45 | 0.60 | 0.02 | 0.02 | 0.02 | 0.02 | 0.03 | 0.83 | 0.79 | 0.71 | 0.53 | 0.37 |

Overall, the learned model exhibited a high level of agreement with the training model. For $n \leq 10$ agreement was 70–90% or higher with the proper choice of $q$. Significantly, we found that the learned model rarely *disagreed* with the training model. As $n$ increases, however, the learned model is increasingly likely to be indecisive about a preference over which the training model is able to reason. For example, for $n = 20$ nodes, we found that agreement ranged from 50 to 60% for $q = 10$, depending on density $\delta$, but disagreement was $< 1\%$. As discussed in Sec. 5.1, this increased indecision as $n$ grows results in part from the use of ordering queries instead of dominance testing as the primary metric.

A question of particular interest was the number of queries per node required. One can observe that an exponential number of outcome comparison queries could be required in the worst case. However, we found that often just a few queries—8 to 14 (even for larger values of $n$)—proved optimal. The choice of $q$ is something of an art. As the data show, increasing the number of queries required to become confident about a node sometimes has an *adverse* effect on the agreement of the learned model with the training model. We take this to be an indication that if $q$ is too high, then overfitting can occur.

We also explored the effect of decreasing the maximum number of parents for the learned model below that of the training model (see Table 3). We found, for example, that for $\mathcal{N}_T$ a maximally dense 7-node graph with maximum 5 parents ($n = 7$, $p = 5$, $\delta = \infty$) and $p$ is set to 5 for $\mathcal{N}_L$, agreement is 90%. If $p$ for $\mathcal{N}_L$

is then reduced to 4, agreement decreases to 70%—still a modestly good result. Furthermore, while indecision is increased, disagreement remained static.

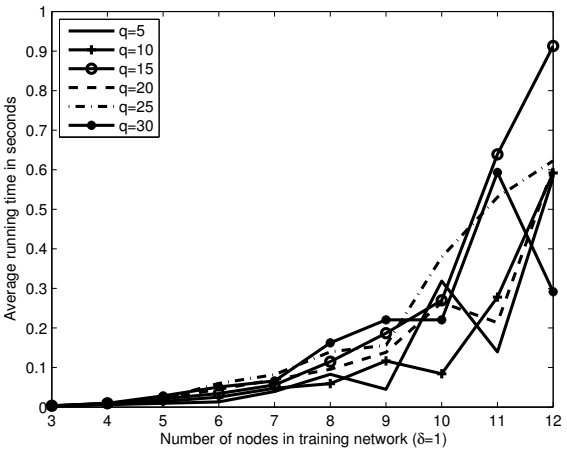


**Fig. 4.** Algorithm Running Times ($\delta = 1$)

We were also interested in the computation time required, since it is known that learning a CP-net is NP-hard in the general case. The relative running times for inputs of various size (Fig. 4 provides data for the case of $\delta = 1$) coincide with our expectations. While the size of the training graph is the primary determinate of running time, the number of queries required by $q$ also plays a role. We also observed that running time can vary significantly from model to model depending on the preference relation that is being learned. Notice, also, that for some $q$ values, time complexity does not grow monotonically. When we generate queries in order to learn the CPT for $v_i$, those queries may be relevant to other nodes $v_j$ in the *Unconfident* set. It may be that, when we come to $v_j$, we already have $q$ many relevant comparisons.

## 6 Summary and Future Research

We have presented an algorithm for learning CP-nets from queries that is efficient and is guaranteed to produce output. Our tests show that the output CP-nets are close approximations to the underlying CP-nets used to generate answers to queries, particularly if we have a close match between the parameters $\delta$ and $p$.

The next step in our research plan is to extend our algorithm to handle noisy and possibly inconsistent responses to queries.

## 7 Acknowledgments

We would like to thank the anonymous referees for their valuable comments. This material is based upon work supported by the National Science Foundation under Grants No. CCF-1215985 and CCF-1049360. The content reflects the views of the authors and not necessarily those of NSF.

## References

1. Boutilier, C., Brafman, R.I., Hoos, H.H., Poole, D.: Reasoning with conditional ceteris paribus preference statements. In: UAI-99. (1999) 71–80
2. Lang, J., Mengin, J.: Learning preference relations over combinatorial domains. In: NMR-08. (2008)
3. Lang, J., Mengin, J.: The complexity of learning separable ceteris paribus preferences. In: IJCAI-09, San Francisco, CA, USA, Morgan Kaufmann. (2009) 848–853
4. Eckhardt, A., Vojtáš, P.: How to learn fuzzy user preferences with variable objectives. In: IFSA/EUSFLAT. (2009) 938–943
5. Eckhardt, A., Vojtáš, P.: Learning user preferences for 2CP-regression for a recommender system. In: SOFSEM-10. (2010) 346–357
6. Koriche, F., Zanuttini, B.: Learning conditional preference networks with queries. In: IJCAI-09. (2009) 1930–1935
7. Koriche, F., Zanuttini, B.: Learning conditional preference networks. Artificial Intelligence **174** (2010) 685–703
8. Dimopoulos, Y., Michael, L., Athienitou, F.: Ceteris paribus preference elicitation with predictive guarantees. In: IJCAI-09, San Francisco, CA, USA, Morgan Kaufmann (2009) 1890–1895
9. Liu, J., Xiong, Y., Wu, C., Yao, Z., Liu, W.: Learning conditional preference networks from inconsistent examples. TKDE **PP**(99) (2012) 1
10. Knuth, D.E.: The Art of Computer Programming. Volume 4A. Addison–Wesley (1997)
11. Goldsmith, J., Lang, J., Truszczyński, M., Wilson, N.: The computational complexity of dominance and consistency in CP-nets. JAIR **33** (November 2008) 403–432
12. Guerin, J.T.: Graphical Models for Decision Support in Academic Advising. PhD thesis, University of Kentucky (2012)