Maintaining Dynamic COP-nets with Changing Preferences

Ki Hyang Lee[1], Michael Fleming[1], Scott Buffett[2]

1. Faculty of Computer Science, University of New Brunswick *2. National Research Council of Canada*

Introduction

Preference elicitation and reasoning with these preferences have been studied for the automated decision making process, to help to easily determine whether a user prefers one outcome over another and to estimate utilities for the entire set of outcomes, based on the assumption that a user's preferences are stable. A Conditional Outcome Preference Network (COP-net) [1] is a graphical structure for representing user preferences and for predicting preferences over all feasible outcomes. In our research, we provide for situations in which users might have ***changing*** preferences. ***Dynamic*** COP-nets are maintained, using an approach that does not require the entire preference model to be rebuilt when a previously-learned preference is changed.

Problem Statement

The first problem is the usual assumption of preference stability. The goal is to modify the preference model whenever there is a changed preference without having to rebuild the initial preference structure.

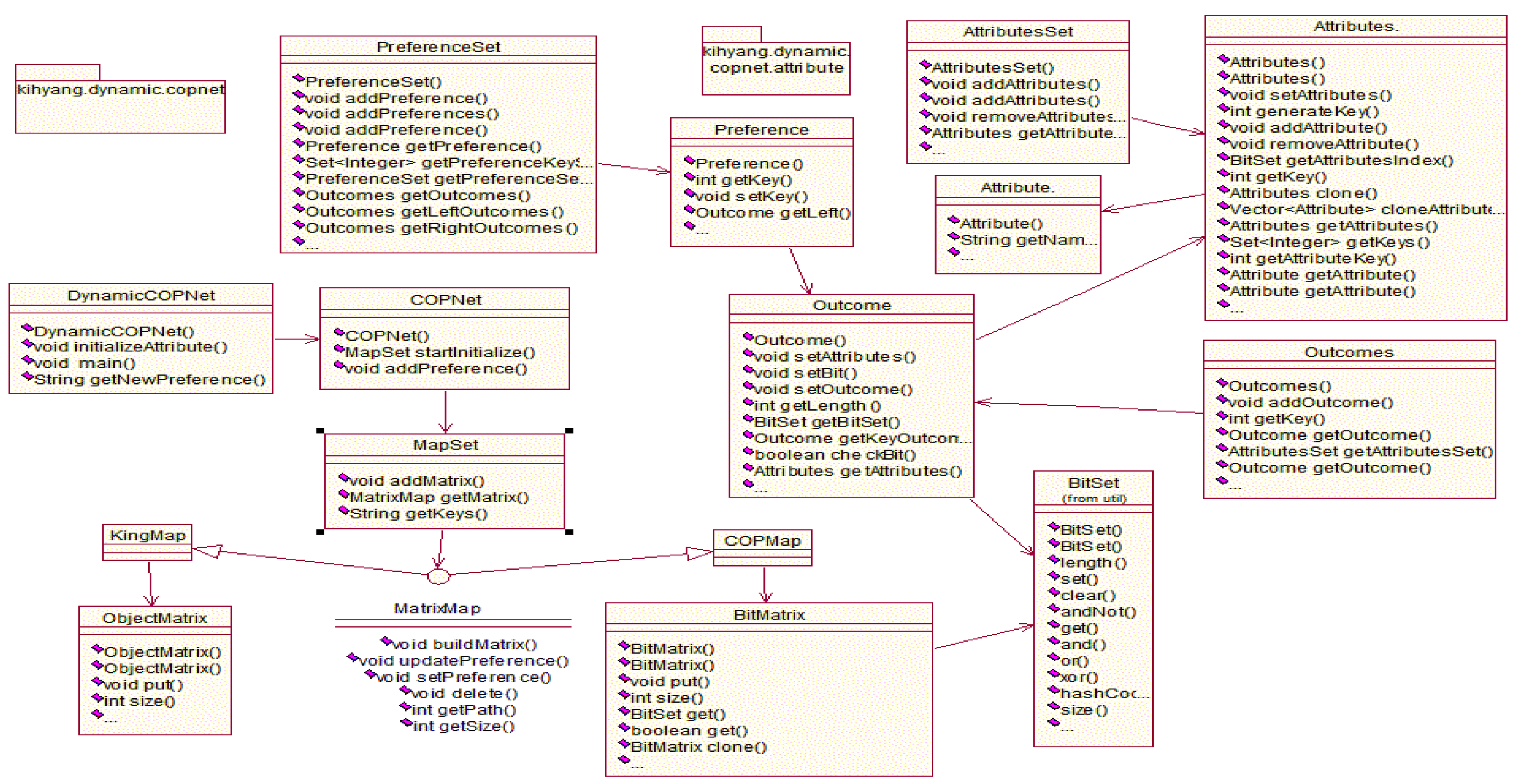
The second problem has to do with the need for the changing preference graph to remain ***transitively reduced***, meaning that there are no redundant edges. The goal is to provide an efficient algorithm that maintains the transitive reduction in $O(n^2)$ and can answer the question "Is outcome o_1 preferred over o_2?" in O(1) time. The table below shows the result of King's algorithm [2] with a worst-case time complexity of $O(n^2)$ with O(1) query time; it still has some deficiencies in that it requires another path matrix for information that is not covered by the adjacency matrix.

Number of all feasible outcomes	Average insert time (ms)	Average delete time (ms)
64	0	1
128	8	31
512	6	110
2048	95	244
8192	3082	5040

Proposed Solution

1. Dynamic COP-net Framework

In order to provide efficiency and flexibility for Dynamic COP-nets, we designed our framework based on components such as Outcomes, Attributes, Preferences, BitSet and BitMatrix to support proper functions.



2. Maintaining Transitive Reduction algorithm

In order to provide efficient maintenance of the transitive reduction in $O(n^2)$ time with O(1) query time, there are 2 adjacency matrices. A matrix M_1 stores edges representing a set of preferences elicited from a user, and matrix M_2 is for a set of distinct directed paths with length greater than 1.

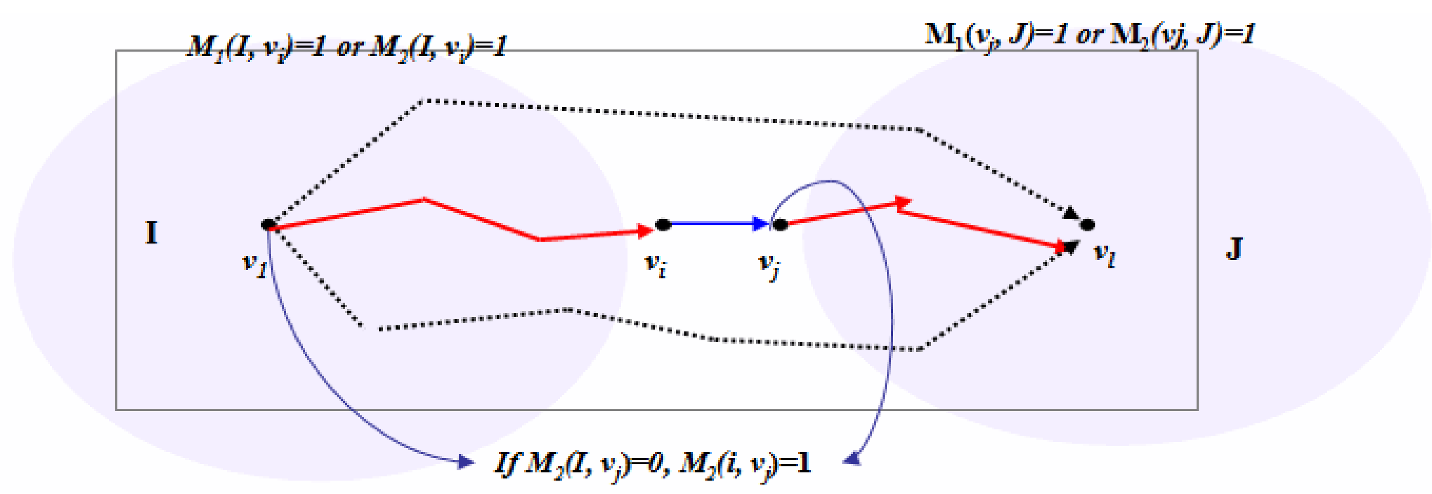
Proposed Solution (Cont.)

There are 4 conditions between two vertices *i* and *j* in the reduced COP-net.

Condition 1. If there is no path from i to j, then the value of (i, j) is 0.
Condition 2. If there is an edge (i, j) but no other directed path, then the value of (i, j) is 1.
Condition 3. If there is no edge (i, j) but a directed path from i to j, then the value of (i, j) is 2.
Condition 4. If there is an edge (i, j) and a directed path, then the value of (i, j) is 3.

A COP-network is a directed acyclic graph.
Let G=<V, E> be a directed acyclic graph representing a COP-network.
The adjacency matrices M_1 and M_2 of G are the $|V| \times |V|$ matrices with values {0,1}.
Building an initial adjacency matrices M_1, M_2
1. $M_1(i, j)$, $M_2(i, j) = 0$ for every $i, j \in V$.
2. Update $M_1(i, j) = 1$, if $(i, j) \in E$ for $i, j \in V$.
3. Add $M_2(i, j)=1$, if there is a directed path of length greater than 1 from *i* to *j* for $M_2(i, j)=0$, $i \neq j$ and $i, j \in V$.

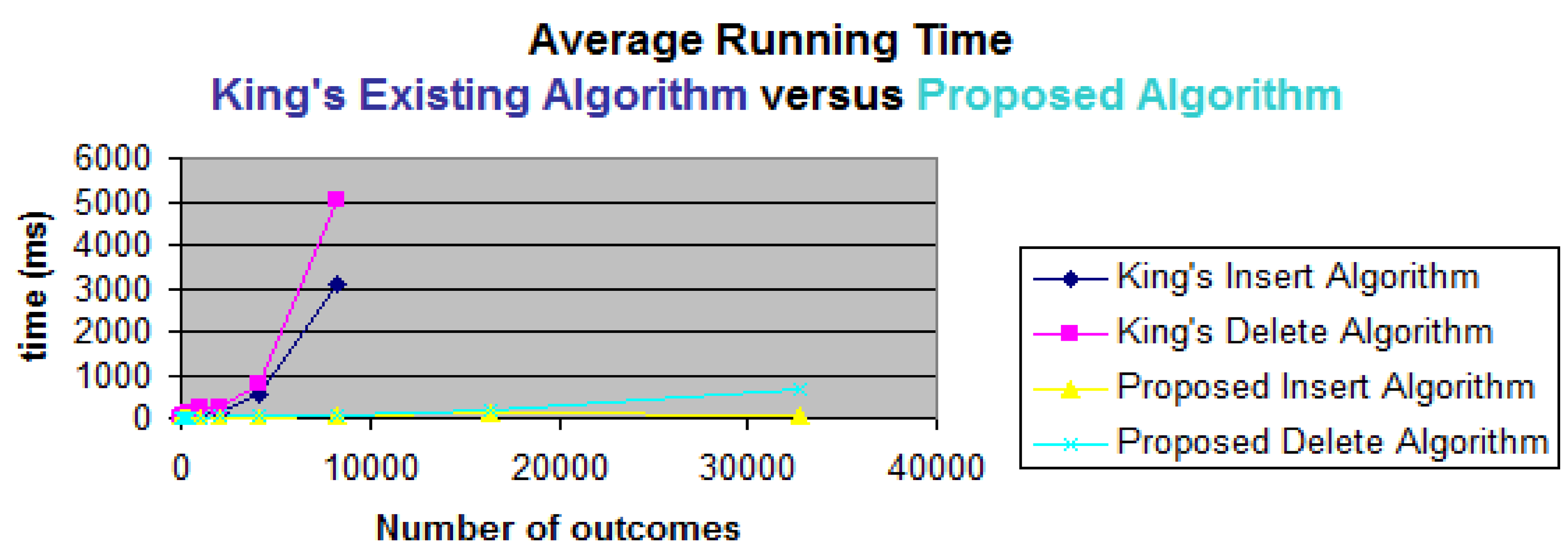
Finally, a reduced COP-network can be represented by every vertex *i, j* where $M_1(i, j) = 1$ and $M_2(i, j)=0$.



For inserting a new edge (v_i, v_j) on the directed acyclic graph, there are three steps.
The first step is for all $I \in V$ such that $M_1(I, v_i)=1$ or $M_2(I, v_i)=1$, if $M_2(I, v_j)=0$, then $M_2(I, v_j) = 1$.
The second step is for all $J \in V$ such that $M_1(v_j, J) =1$ or $M_2(v_j, J)=1$, if $M_2(v_i, J)=0$, then $M_2(v_i, J) = 1$.
The third step is for all $I, J \in V$ such that ($M_1(I, v_i)=1$ or $M_2(I, v_i)=1$) and ($M_1(v_j, J)=1$ or $M_2(v_j, J)=1$), if $M_2(I, J)=0$, then $M_2(I, J)=1$.
For the deletion of an edge (v_i, v_j), for all $I, J \in V$ such that ($M_1(I, v_i)=1$ or $M_2(I, v_i)=1$) and ($M_1(v_j, J)=1$ or $M_2(v_j, J)=1$), if there is a vertex *m* such that ($M_1(m, v_i)=0$ and $M_2(m, v_i)=0$) and ($M_1(v_j, m)=0$ and $M_2(v_j, m)=0$) and if ($M_1(I, m)=1$ or $M_2(I, m)=1$) and ($M_1(m, J)=1$ or $M_2(m, j)=1$) then there is a path from *I* to *J* not containing (v_i, v_j).

Current Results

The chart below shows the current results comparing King's existing algorithm to the proposed algorithm, with the average running time for insertion and deletion. It shows that our proposed insertion and deletion algorithms have reduced their running time as well as increased the number of outcomes that can be maintained.



References

[1] S. Chen, S. Buffett, and M. Fleming, "Reasoning with Conditional Preferences Across Attributes" in 20th Canadian Conference on Artificial Intelligence, pp. 369-380, 2007.

[2] V. King and G. Sagert, "A Fully Dynamic Algorithm for Maintaining the Transitive Closure", Journal of Computer and System Sciences 65, pp.150–167, 2002.

Faculty of Computer Science UNB

National Research Council Canada Conseil national de recherches Canada