A Calculus of Partially Ordered Preferences for Compositional Modelling and Configuration

Jeroen Keppens and Qiang Shen
Centre for Intelligent Systems and their Applications
The University of Edinburgh
{jeroen,qiangs}@dai.ed.ac.uk

Abstract

Preference elicitation to support solving synthesis problems in certain domains (e.g. automated ecological model construction) is inhibited by a severe lack of knowledge about the criteria that motivate decision making. Yet, even in these domains, humans are able to provide some partial ordering of their preferences, based on past experience and personal opinion. Working towards an efficient representation and reasoning mechanism with such partial preference information, this paper introduces a qualitative calculus of partially ordered preferences that is rooted in order of magnitude reasoning. It then integrates this calculus in a dynamic constraint satisfaction problem. A solution algorithm for the resulting dynamic preference constraint satisfaction problem is also presented. To demonstrate the ideas, the proposed techniques are applied to sample compositional modelling and configuration tasks.

Introduction

Many synthesis problems, such as configuration and compositional modelling (Falkenhainer, B. & Forbus, K.D. 1991; Keppens, J. & Shen, Q. 2001) occur in a setting that is both constrained by hard requirements and affected by the preferences of a decision maker. Often, the assembly of an artifact or a model must obey certain physical/mathematical laws and it must meet the purpose for which it is being created. In the meantime, the person who is to use the artifact or model may have personal preferences with respect to the available artifact/model design decisions. This must also be considered.

Dynamic constraint satisfaction problems (DCSPs) and the corresponding solution algorithms (Mittal, S. & Falkenhainer, B. 1990) are widely used in to solve configuration problems (Birmingham, W.P., Gupta, A.P., & Siewiorek, D.P. 1992) and compositional modelling tasks (Falkenhainer, B. & Forbus, K.D. 1991; Keppens, J. & Shen, Q. 2000). However, approaches to integrate a DCSP with preferences to form a dynamic preference constraint satisfaction problem (DPCSP) are limited. The ones that do exist rely on idempotent operators ($p \oplus p = p$) to combine preference valuations, thereby ignoring the additive utilities or preferences derived from independent features in an artifact or model.

In addition, methods for expressing and comparing preference valuations in a way that is suitable for compositional modelling and configuration tasks is an ongoing research issue. In conventional decision theory, the possible outcomes of decisions are assigned a utility or preference taken from a totally ordered domain. Then, the decision maker considers the available options and selects the one that maximises the expected utility that follows from the corresponding decision (Doyle, J. & Thomason, H. 1999). Such an approach is desirable when a prescriptive rational decision maker is to be implemented, for example in an agent system. However, human decision-making rarely follows this pattern (Tversky, A. & Thaler, R.H. 1990).

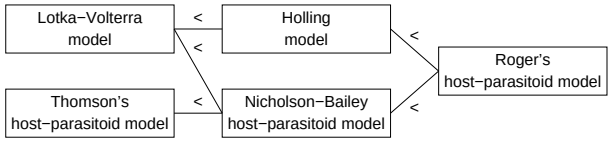

Figure 1: Sample partial preference ordering

Current research addresses this by devising logics that help model the decision making process (Delgrande, J.P. & Schaub, T. 2000; Brewka, G., Benferhat, S., & Le Berre, D. 2002) and preference elicitation methods (Ha, V. & Haddawy, P. 1997). In certain domains, however, it is very difficult to derive a rational reasoning process or to elicit the criteria underlying a person's preferences. In such cases, a more direct approach is required to represent, combine and compare partially ordered preferences.

Consider, for instance, the problem of selecting an approach to describe the interaction between a host population and a parasitoid population in ecological modelling. Knowledge on how to model this phenomenon is incomplete, and it is difficult to identify what knowledge is missing. Important approaches, such as using a causal model that links partial DPCSP solutions to desirable features (Boutilier, C. *et al.* 1997), are not applicable in this domain because ecologists are often in disagreement about the primitive features and the initial causal model. Yet, ecologists may have subjective preference orderings over the modelling choices that can be made. For example, figure 1 presents a sample preference ordering that some ecologists may agree with and others may disagree with. As ecologists have different preference orderings, they tend to build different models to match these preferences.

In this paper, a preference calculus, rooted in order of magnitude reasoning (Raiman 1991), is introduced to conveniently describe preference orderings and to efficiently compare the preferences. This calculus is then incorporated into the framework of DPCSP and a basic solution algorithm is presented to solve such problems. Its use is illustrated by means of sample compositional modelling and configuration problems.

Background

A classical hard CSP is specified by

- a set of attributes $\mathbf{X} = \{x_1, \ldots, x_n\}$,

- a set of domains $\mathbf{D} = \{D_1, \ldots, D_n\}$ with $D_i = \{d_{i1}, \ldots, d_{in_i}\}$ for each attribute x_i, and
- a set of compatibility constraints $\mathbf{C}$, where a compatibility constraint c over attributes $x_i, \ldots, x_j$ is a relation $c : D_i \times \ldots \times D_j \rightarrow \{\top, \bot\}$.

A set of assignments $\{x_1 : d_{1k_1}, \ldots, x_i : d_{ik_i}, \ldots, x_j : d_{jk_j}, \ldots, x_n : d_{nk_n}\}$ is said to satisfy this compatibility constraint c if $c(d_{ik_i}, \ldots, d_{jk_j}) = \top$.

A DCSP, as defined in (Mittal, S. & Falkenhainer, B. 1990), is an extension of a hard CSP in which attributes can be active and inactive. An attribute x_i is said to be active (denoted by active(x_i)) if and only if it is assigned a value from its domain. The activity of attributes is governed by a set of activity constraints $\mathbf{A}$, which are defined via implications that establish conditions under which certain attributes become active. A set of assignments $\{x_1 : d_{1k_1}, \ldots, x_m : d_{mk_k}, \neg\text{active}(x_{m+1}), \ldots, \neg\text{active}(x_n)\}$ is said to satisfy an activity constraint a if the conjunction of assignments is not inconsistent with a, that is $(x_1 : d_{1k_1} \wedge \ldots \wedge x_m : d_{mk_k}, \neg\text{active}(x_{m+1}) \wedge \ldots \wedge \neg\text{active}(x_n)), a \nvdash \bot$.

Configuration as a DCSP

According to (Mittal, S. & Frayman, F. 1989), a configuration task is described by a specification of the desired properties of the configuration and by a set of components, where each component is defined by (1) a number of ports that connect it to other components, (2) a set of constraints at each port restricting the components that can be connected to it, and (3) other structural constraints. Such a task can be translated into a DCSP:

- The components normally be grouped in component classes, such that the components that belong to the same class perform a particular function (in a different way) and they have certain ports in common that connect it to the rest of the system. As such, in the DCSP, each component class corresponds to an attribute $x_i \in \mathbf{X}$ and defines the domain $D_i = \{d_{i1}, \ldots, d_{in_i}\}$ of that attribute.
- The ports that the components connect to may be part of every system, or they may become available as they are introduced by other components. Such relations are translated into activity constraints. For example, in a car configuration problem, the choice of airconditioner is only active if the car's engine can support airconditioning.
- The structural constraints in the system govern the combinations of components that can be connected to certain ports. Such restrictions are translated into compatibility constraints.

Compositional Modelling as a DCSP

According to (Keppens, J. & Shen, Q. 2001), a compositional modelling task is described by a scenario, a domain theory and certain requirements that specify what makes models of the scenario adequate. A scenario is a high level specification of a system of interest, such as a component-connection diagram. A domain theory consists of a set of translation methods, called model fragments, that turn descriptions of components and processes into more detailed descriptions of these components and processes. A compositional modeller aims to apply a combination of these model fragments onto the scenario in order to turn it into a scenario model that meets the adequacy requirements. The problem of finding the right combination of model fragments can be posed as a DCSP:

- The attributes of the DCSP correspond to so-called assumption classes. Assumption class is the overloaded concept used in compositional modelling to describe exhaustive sets of mutually exclusive elements on the basis of which modelling decisions are made. In some approaches, assumption classes contain different model fragments describing alternative models of the same component or process (Nayak, P.P. & Joskowicz, L. 1996; Levy, A.Y., Iwasaki, Y., & Fikes, R. 1997). In other approaches, assumption classes are design decisions that the modeller must make. In this case, the assumption classes are sets of alternative assumptions that may underpin a model (Falkenhainer, B. & Forbus, K.D. 1991; Keppens, J. & Shen, Q. 2000). Thus, when translating a compositional modelling problem into a DCSP, each assumption class corresponds to an attribute and the contents of the assumption class form the attribute's domain.
- Similar to the component classes in the configuration task, there may be activity constraints that govern the conditions under which an assumption class is part of the solution to the compositional modelling task. For example, in a compositional ecological modelling problem, an attribute denoting the model for population growth is only active under the conditions where an a variable describing population size exists.
- The model representation formalism and the requirements imposed upon the adequacy of the scenario model restrict the combinations of assumptions/model fragments that are consistent. For example, in a systems dynamics model of an ecological system, two different equations that compute population births in a different way can not be combined in the same model. Such restrictions are translated into compatibility constraints.

Dynamic preference constraint satisfaction

The present work enriches the notion of DCSP, allowing the representation and thereforth the solution of dynamic preference constraint satisfaction problems (DPCSPs), by introducing to a DCSP elements from valued constraint satisfaction (Schiex, T., Fargier, H., & Verfaillie, G. 1995). More formally, a DPCSP extends a DCSP with a preference valuation $p(x_i : d_{ij}) \in \mathbb{P}$ for each attribute-value assignment $x_i : d_{ij}$, where $\mathbb{P}$ denotes the domain of preference valuations. The preference of a (partial) solution $\{x_i : d_{ik_i}, \ldots, x_j : d_{jk_j}\}$ is computed as

$$p(x_i : d_{ik_i}, \ldots, x_j : d_{jk_j}) = p(x_i : d_{ik_i}) \oplus \ldots \oplus p(x_j : d_{jk_j})$$

where $\oplus$ is a commutative, associative, closed binary operation on $\mathbb{P}$. The preference values in $\mathbb{P}$ are partially ordered by $\prec$, where $p_i \prec p_j$ (with $p_i, p_j \in \mathbb{P}$) is interpreted so that the assignment associated with p_j has a higher preference over the assignment associated with p_i.

This use of preferences differs from other work employing a qualitative preference calculus to solve synthesis problems, such as (Boutilier, C. *et al.* 1997). In the latter, partial overall preference orderings stem from preference orderings provided over different feature attributes. Such an approach is well suited for a domain where a generally agreed causal model exists, linking attribute value assignments to features. This work focuses on problems where no such causal model is available.

The solution of such a CSP consists of all sets of assignments $\{x_i : d_{ik_i}, \ldots, x_j : d_{jk_j}\}$ that satisfy all given compatibility and activity constraints, such that no other sets of assignments $\{x_p : d_{pk_p}, \ldots, x_q : d_{qk_q}\}$ satisfy the compatibility and activity constraints with $p(x_i : d_{ik_i}, \ldots, x_j : d_{jk_j}) \prec p(x_p : d_{pk_p}, \ldots, x_q : d_{qk_q})$. In the context of configuration and compositional modelling tasks, the preferences describe the utility contributions of the various components or modelling choices. As such, the optimal solution to a DPCSP, corresponds to a preferred configuration or scenario model of a system.

DPCSPs differ from existing types of CSP in two respects. Firstly, they integrate the features of *dynamic* CSPs with those of *valued* CSPs, thus providing a richer representational framework. Secondly, unlike the vast majority of valued CSPs, which are types of so-called semiring-based CSPs (Bistarelli, S., Montanari, U., & Rossi, F. 1997), the preference combination operator $\oplus$ employed in this work is not assumed to be idempotent ($a \oplus a = a$). Idempotent combination operators are commonly employed in valued CSPs because they enable the use of existing local consistency algorithms (Schiex, T., Fargier, H., & Verfaillie, G. 1995), which are known to be effective and efficient. However, the semantics of idempotent combination operators are not always suitable for synthesis problems. In DPCSPs, the preferences express utility contributions of individual attribute-value assignments, and each of these utility contributions is presumed to add to the overall utility (and therefore $a \oplus a$ should be preferred over a, rather than $a \oplus a = a$).

Order of magnitude preference calculus

In DPCSPs, valuations are attached to individual attribute assignments and are combined to obtain a preference value for an emerging CSP solution. In poorly understood domains, the knowledge about such valuations is often very limited. In particular, a direct comparison between preference values is usually not possible as they are not totally ordered. Therefore, this work considers partial orderings of preferences, such as the one shown in figure 1, and priorities amongst the preferences that need to be optimised.

Order of magnitude reasoning (OMR) (Raiman 1991) provides a well-established framework for symbolic calculi to reason about qualitative distinctions between quantities at different levels of granularity. However, all existing OMR calculi assume that the underlying domain of quantities is totally ordered. Thus, this section introduces a novel OMR calculus to reason about partially ordered space of quantities.

Theoretical foundation

In this work, it is presumed that the user specifies a space $\mathbb{B}$ of basic preference quantities (BPQs). BPQs are the smallest units of preference valuation and are partially ordered. BPQs are related to one another by the "order of magnitude smaller than" relation $\ll$, the "equivalent order of magnitude as" relation $\sim$ and by the "smaller than within the same order of magnitude" relation $<$. Note that the latter relation also implies that the BPQs have an equivalent order of magnitude. Therefore, $\forall p_1, p_2 \in \mathbb{B}, p_1 < p_2 \rightarrow p_1 \sim p_2$. Naturally, order of magnitude smaller than relations are shared by all BPQs within the same order of magnitude. That is,

$$\forall p_1, p_2, p_3 \in \mathbb{B}, p_1 \sim p_2 \wedge p_2 \ll p_3 \rightarrow p_1 \ll p_3$$
$$\forall p_1, p_2, p_3 \in \mathbb{B}, p_1 \sim p_2 \wedge p_3 \ll p_2 \rightarrow p_3 \ll p_1$$

BPQs are combined with one another to form so-called order of magnitude preferences (OMPs). In general, the implicit value of an OMP P equals the combination $p_1 \oplus \ldots \oplus p_n$ of its constituent BPQs $p_1, \ldots, p_n$. In what follows, an approach will be presented to compute a partial ordering relation $\prec$ over the OMPs, based on the constituent BPQs of the OMPs. Generally speaking, the calculus is based on the following assumptions:

- *Properties of* $\oplus$: The combination operator $\oplus$ is assumed to be commutative, associative and strictly monotonic ($P \prec P \oplus P$). The latter assumption is made to better reflect the ideas underpinning conventional utility calculi.
- *Prioritisation*: A combination of BPQs is never an order of magnitude greater than its constituent BPQs. That is, given the following ordering of BPQs $p_1 \sim p_2 \sim \ldots \sim p_n \ll p$, then
  $$p_1 \oplus p_2 \oplus \ldots \oplus p_n \prec p$$
  Also, distinctions at higher orders of magnitude are considered to be more significant than those at lower orders of magnitude. That is, given an ordering of BPQs $p_1 \sim \ldots \sim p_{m-1} \sim p_m \sim \ldots \sim p_n \ll p_a < p_b$, then
  $$p_1 \oplus \ldots \oplus p_{m-1} \oplus p_a \prec p_m \oplus \ldots \oplus p_n \oplus p_b$$
  In terms of OMPs, it means that the DPCSP algorithm will prioritise the optimisation associated with preferences of higher order of magnitude.
- *Strict monotonicity*: Even though distinctions at higher orders of magnitude are more significant, distinctions at lower orders of magnitude are not negligible. That is, given an ordering of BPQs $p_1 < p_2$ and an OMP P, then $p_1 \oplus P \prec p_2 \oplus P$, irrespective of the orders of magnitude of the BPQs that constitute P. This is a departure from conventional OMR. If the OMPs associated with two (partial) DPCSP solutions contain equal BPQs at a higher order of magnitude, it is usually desirable to compare both solutions further in terms of the (less important) constituent BPQs at lower orders of magnitude.
- *Partial ordering maintenance*: Conventional OMR is motivated by the need for abstract descriptions of real-world behaviour, whereas the OMP calculus is motivated by incomplete information. As opposed to conventional OMR, OMPs do not map onto the real number line. This implies that, when the user states, for example, that $p_1 < p_2 < p$ and that $p_3 < p_4 < p$, the explicit absence of ordering information between the BPQs in $\{p_1, p_2\}$ and those in $\{p_3, p_4\}$ means that the user can not compare them (e.g. because they are entirely different things). Consequently, $p_1 \oplus p_2$ would be deemed incomparable to $p_3 \oplus p_4$ (i.e. $p_1 \oplus p_2 ? p_3 \oplus p_4$), rather than roughly equivalent.

These assumptions can be formalised in a more general definition of the ordering relation $\prec$. Let an OMP $P = p_1 \oplus \ldots \oplus p_n$ be defined as a function $f_P : \mathbb{B} \rightarrow \mathbb{N} : p \mapsto f_P(p)$ where $\mathbb{B}$ is the set of BPQs, $\mathbb{N}$ is the set of natural numbers and $f_P(p)$ calculates the number of occurrences of p in $p_1, \ldots, p_n$. For example, given that $P = p_a \oplus p_b \oplus p_b$, then $f_P(p_a) = 1$ and $f_P(p_b) = 2$. Let $\mathbb{B}(p)$, $p \in \mathbb{B}$, be the subset of $\mathbb{B}$ that contains the BPQs of the same order of magnitude as p, i.e. $\mathbb{B}(p) = \{p_i \mid p_i \in \mathbb{B}, p_i \sim p\}$. Then, the constituent BPQs of an OMP P_1 that are within the same order of magnitude as a given BPQ p, are less than or equal to those of an OMP P_2 if $\forall p_i \in \mathbb{B}(p)$:

$$(f_{P_1}(p_i) + \sum_{p_j \in \mathbb{B}, p_i < p_j} f_{P_1}(p_j)) \leq (f_{P_2}(p_i) + \sum_{p_j \in \mathbb{B}, p_i < p_j} f_{P_2}(p_j))$$

This is denoted by $P_1 \preccurlyeq_p P_2$. The constituent BPQs of an OMP P_1 that are within the same order of magnitude as a given BPQ p, are less than but not equal to those of an OMP P_2 if $P_1 \preccurlyeq_p P_2 \wedge \neg(P_2 \preccurlyeq_p P_1)$. This is denoted by $P_1 \prec_p P_2$. More generally, an OMP P_1 is less than an OMP P_2 if, for each distinct order of magnitude, either P_1 is less than P_2 for the BPQs within this order of magnitude, or there are BPQs at a higher order of magnitude for which P_1 is less than P_2:

$$P_1 \prec P_2 \leftarrow \forall p_a \in \mathbb{B}, (P_1 \prec_{p_a} P_2) \vee (\exists p_b \in \mathbb{B}, p_a \ll p_b \wedge P_1 \prec_{p_b} P_2)$$

It can be shown that this definition of $\prec$ results in a partial ordering of OMPs that meets the aforementioned assumptions. It allows for the case where two OMPs P_1 and P_2 are incomparable, denoted by $P_1?P_2$, meaning $\neg(P_1 \prec P_2) \wedge \neg(P_2 \prec P_1) \wedge (P_1 \neq P_2)$ holds. Note that $P_1 = P_2$ if P_1 and P_2 are a combination of the same collection of BPQs.

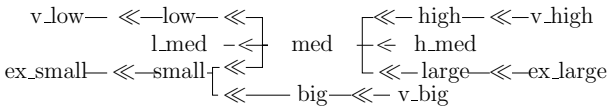

Figure 2: Sample OM preference scale

Figure 2 shows a sample set of BPQs and ordering relations between them. Based on it, the following comparisons can be made between OMPs:

$$\text{h\_med} \oplus \text{med} \prec \text{high} \quad (1)$$

$$\text{h\_med} \oplus \text{med} \succ \text{med} \oplus \text{l-med} \quad (2)$$

$$\text{high} \oplus \text{large?big} \oplus \text{small} \quad (3)$$

Relation (1) is true because high is an order of magnitude greater than h_med and med. Relation (2) is justified by (h_med > med) and (med > l_med). The OMPs in (3) are incomparable because there is no path between big and high or between big and large.

Comparison of order of magnitude preferences

To show the decidability of the above theory, this subsection describes an algorithm to compare two OMPs. First, some further definitions must be introduced.

- The ordering relations $<$ and $\ll$, which are responsible for the ordering of BPQs are redefined for presentational simplicity as sets of pairs:

$$O_< = \{(p_1, p_2) \mid p_1 < p_2\}, \quad O_\ll = \{(p_1, p_2) \mid p_1 \ll p_2\}$$

- A *cross-over quantity* p with respect to an ordering relation $O \in \{O_<, O_\ll\}$, which is expressed by $\text{co}(p, O)$, is a BPQ for which O defines at least two BPQs to be greater or smaller than it. That is:

$$\forall p, \exists p_1, \exists p_2, p_1 \neq p_2, ((p_1, p) \in O \wedge (p_2, p) \in O) \vee ((p, p_1) \in O \wedge (p, p_2) \in O) \rightarrow \text{co}(p, O)$$

In the ongoing example, small and med are the two cross-over quantities.

- A set of BPQs $\{p_1, p_2, \ldots, p_n\}$ is regarded as a *path* from p_b to p_e with respect to an ordering relation O, denoted by $\text{path}(O, p_b, p_e)$, if $(p_b, p_1) \in O$, $(p_1, p_2) \in O$, ..., $(p_n, p_e) \in O$. For example, $\{\text{med}, \text{high}\}$ is a $\text{path}(O_\ll, \text{small}, \text{v\_high})$.

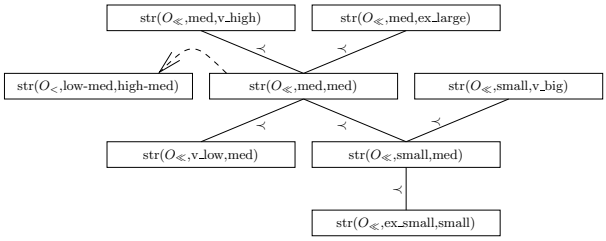

Figure 3: Ordering of strands

- The *distance* of a path equals the number of BPQs in it. That is, the distance $d(O, p_b, p_e)$ between two BPQs p_b and p_e with respect to an ordering relation O equals the largest distance of any paths between p_b and p_e. When there is a path from p_e to p_b, then $d(O, p_e, p_b) = -d(O, p_b, p_e)$. For example, $d(O_\ll, \text{small}, \text{v\_high}) = 2$.
- The space of BPQs $\mathbb{B}$ is partitioned into totally ordered subsets of BPQs. That is, given an ordering relation O, a strand $\text{str}(O, p_1, p_2)$ is a $\text{path}(O, p_1, p_2)$ such that

$$(\text{co}(p_1, O) \vee \nexists p' \in \mathbb{B}, (p', p_1) \in O) \wedge$$
$$(\text{co}(p_2, O) \vee \nexists p' \in \mathbb{B}, (p_2, p') \in O)$$

and that $\text{path}(O, p_1, p_2)$ does not contain any cross-over quantities. Additionally, each cross-over quantity p_c is said to be the only member of the strand $\text{str}(O, p_c, p_c)$. It can be shown that the strands defined by an ordering relation entail a unique partition of $\mathbb{B}$.

The strands can be compared with one another based on the BPQs within them. A strand s_1 is smaller than a strand s_2, denoted $s_1 < s_2$, if:

$$\forall p_1, p_2 \in \mathbb{B}, p_1 \in s_1 \wedge p_2 \in s_2 \rightarrow \text{path}(O, p_1, p_2)$$

Figure 3 shows the strands in the ongoing example and their ordering.

Each BPQ $p \in \mathbb{B}$ can now be uniquely identified by the strand $\text{str}(O, p_1, p_2)$ of which it is a member and by the distance $d(O, p_1, p)$. The tuple $\langle \text{str}(O, p_1, p_2), d(O, p_1, p) \rangle$ is said to be the *label* $L(p, O)$ of p within O.

Continuing with the example, the following BPQ labels can be computed:

$$L(\text{small}, O_\ll) = \langle \text{str}(O_\ll, \text{small}, \text{small}), 0 \rangle$$
$$L(\text{big}, O_\ll) = \langle \text{str}(O_\ll, \text{small}, \text{v\_big}), 1 \rangle$$

From this, BPQs can be compared in terms of their labels with respect to an individual ordering O. Given two BPQ labels $L(p_1, O) = \langle s_1, d_1 \rangle$ and $L(p_2, O) = \langle s_2, d_2 \rangle$, $\langle s_1, d_1 \rangle \geq \langle s_2, d_2 \rangle$ if:

$$(s_1 < s_2) \vee (s_1 = s_2 \wedge d_1 \geq d_2)$$

It can be shown that if $L(p_1, O) \geq L(p_2, O)$ a $\text{path}(O, p_1, p_2)$ exists. For instance, in the ongoing example, $L(\text{small}, O_\ll) \leq L(\text{big}, O_\ll)$.

For a given OMP $P = p_1 \oplus \ldots \oplus p_n$, where $p_1, \ldots, p_n \in \mathbb{B}$, a label can be computed by counting the number of each strand-distance pair in the labels of the constituent BPQs:

$$\mathcal{L}(P, O) = \{\langle s, d, n \rangle \mid (n = \sum_{p \in P, L(p,O) = \langle s,d \rangle} 1) \wedge (n \geq 1)\}$$

Algorithm 1: COMPARE(P_1, P_2)

$c_o \leftarrow '='$; $O_1 \leftarrow \mathcal{L}(P_1, O_\ll)$; $O_2 \leftarrow \mathcal{L}(P_2, O_\ll)$;
while $\neg(O_1 = \emptyset \vee O_2 = \emptyset)$
do
- $B_1 \leftarrow \{\langle s,d,n\rangle \in \emptyset_1 \mid \nexists\langle s',d',n'\rangle \in O_1, \langle s,d\rangle < \langle s',d'\rangle\}$;
- $B_2 \leftarrow \{\langle s,d,n\rangle \in \emptyset_2 \mid \nexists\langle s',d',n'\rangle \in O_2, \langle s,d\rangle < \langle s',d'\rangle\}$;
- $H_1 \leftarrow \{\langle s,d,n\rangle \in B_1 \mid \nexists\langle s',d',n'\rangle \in B_2, \langle s,d\rangle \leq \langle s',d'\rangle\}$;
- $H_2 \leftarrow \{\langle s,d,n\rangle \in B_2 \mid \nexists\langle s',d',n'\rangle \in B_1, \langle s,d\rangle \leq \langle s',d'\rangle\}$;
- **if** $(H_1 \neq \emptyset) \wedge (H_2 \neq \emptyset)$
  - **then return** $('?')$;
  - **else**
    - **if** $(H_1 \neq \emptyset)$
      - **then** $c_n \leftarrow '>'$; $E \leftarrow B_1 - H_1$; REMOVE(O_1, H_1); REMOVE(O_2, H_1);
      - **else**
        - **if** $(H_2 \neq \emptyset)$
          - **then** $c_n \leftarrow '\prec'$; $E \leftarrow B_2 - H_2$; REMOVE(O_1, H_2); REMOVE(O_2, H_2);
          - **else** $c_n \leftarrow '='$; $E \leftarrow H_1$;
- **if** $(c_o = c_n) \vee (c_o = '=')$
  - **then**
    - $(c_o, H) \leftarrow$ COMPARE-WM($E, P_1, P_2, O_<, c_n$);
    - **if** $c_o = '?'$
      - **then return** $('?')$;
    - REMOVE(O_1, H); REMOVE(O_2, H);
  - **else return** $('?')$;
- $O_1 \leftarrow O_1 - B_1$; $O_2 \leftarrow O_2 - B_2$;

return (c_o);
procedure COMPARE-WM($E, P_1, P_2, O_<, c_o$)
$H \leftarrow \{\}$;
for each $e \in E$
do
- $L_1 \leftarrow \mathcal{L}(P_1, O_<, \{e\})$; $L_2 \leftarrow \mathcal{L}(P_2, O_<, \{e\})$;
- **if** $\geq_{\text{value}} (L_1, L_2)$
  - **then**
    - **if** $\neg \geq_{\text{value}} (L_2, L_1)$
      - **then**
        - **if** $c_o \in \{'=', '>'\}$
          - **then** $c_o \leftarrow '>'$; $H \leftarrow H \cup \{e\}$;
          - **else return** $('?', H)$;
  - **else**
    - **if** $\geq_{\text{value}} (L_2, L_1)$
      - **then**
        - **if** $c_o \in \{'=', '\prec'\}$
          - **then** $c_o \leftarrow '\prec'$; $H \leftarrow H \cup \{e\}$;
          - **else return** $('?', H)$;
      - **else return** $('?', H)$;

return (c_o, H);

For example,

$$\mathcal{L}(\text{small} \oplus \text{big}, O_\ll) = \{\langle \text{str}(O_\ll, \text{small}, \text{v\_big}), 1, 1\rangle, \langle \text{str}(O_\ll, \text{small}, \text{small}), 0, 1\rangle\}$$

In this way, two labels $\mathcal{L}(P, O_\ll)$ and $\mathcal{L}(P, O_<)$ can be computed for each OMP P. Both labels are related to one another through the underlying set of BPQs they represent. In particular, each strand-distance pair $\langle s_\ll, d_\ll\rangle$ under $O_\ll$ corresponds to a set of strand-distance pairs under $O_<$ that provide finer grain distinctions between quantities. Therefore, a label $\mathcal{L}(P, O_<, L)$ can be obtained, which contains the subset of $\mathcal{L}(P, O_<)$ that corresponds to a subset $L \subset \mathcal{L}(P, O_\ll)$, such that:

$$\mathcal{L}(P, O_<, L) = \{\langle s_<, d_<, n_<\rangle \mid \langle s_<, d_<, n_<\rangle \in \mathcal{L}(P, O_<) \wedge (\exists\langle s_\ll, d_\ll, n_\ll\rangle \in L, \exists q, \langle s_<, d_<\rangle \in \mathcal{L}(q, O_<) \wedge \langle s_\ll, d_\ll\rangle \in \mathcal{L}(q, O_\ll))\}$$

By means of their respective labels, two OMPs can then be compared using the algorithm COMPARE(P_1, P_2). This algorithm iterates through the $O_\ll$ labels of P_1 and P_2. At each iteration, the sets of the highest remaining strand-distance pairs in the labels of P_1 and P_2, respectively denoted B_1 and B_2, are considered. The algorithm systematically compares the tuples $\langle s,d,n\rangle$ of B_1 and B_2 with one another:

- Each $\langle s,d,n\rangle \in B_1$ ($\langle s,d,n\rangle \in B_2$), such that $\langle s,d,n\rangle$ is either greater than or incomparable to the tuples in B_2 (B_1), is stored in a set H_1 (H_2), with $H_1 \neq \emptyset$ ($H_2 \neq \emptyset$) indicating that $P_1 \succ P_2$ ($P_2 \succ P_1$) for the tuples involved, i.e. $\forall p \in H_1, P_2 \prec_p P_1$ ($\forall p \in H_2, P_1 \prec_p P_2$).
- The tuples $\langle s,d,n\rangle \in B_1$ ($\langle s,d,n\rangle \in B_2$), such that a tuple $\langle s,d,n'\rangle \in B_2$ ($\langle s,d,n'\rangle \in B_1$) exists are stored in a set E.

Because the BPQs referred by the tuples in E can not be compared with one another by means of the $O_\ll$ labels, the corresponding $O_<$ labels L_1 and L_2 are computed and compared in COMPARE-WM(). In order to determine whether L_1 is greater than L_2, this procedure employs the $\geq_{\text{value}}$ function:

$$\geq_{\text{value}} (L_1, L_2) \leftarrow \forall\langle s_2, d_2, n_2\rangle \in L_2, \sum_{\langle s,d,n\rangle \in L_1, \langle s_2,d_2\rangle \leq \langle s,d\rangle} n \geq \sum_{\langle s,d,n\rangle \in L_2, \langle s_2,d_2\rangle \leq \langle s,d\rangle} n$$

The results of these comparisons may be contradictory when at least one tuple from B_1 is greater than anything in B_2 and at least one tuple from B_2 is greater than anything in B_1. In that case, the algorithm terminates with returning incomparable ($'?'$). If the comparisons are not contradictory, then the tuples in B_1 and B_2 which yielded a strict ordering $'\prec'$ or $'\succ'$ (i.e. H_1 or H_2), need to be differentiated from the ones for which no such ordering could be established. All tuples in the labels of P_1 and P_2 that are smaller than H_1 or H_2 are removed. For the remaining tuples, the algorithm considers the next highest ones, that are still part of the labels, in the subsequent iteration or it terminates if there are no remaining tuples. The variables c_o and c_n are used in the algorithm to store the outcome of partial comparisons with respect to BPQs in other parts of the partial ordering.

For example, when applying this algorithm to compare P_1 = big $\oplus$ small with P_2 = high $\oplus$ large, the highest strand-distance pairs from the $O_\ll$ labels are compared first. This leads to $H_1 = \{\langle \text{str}(O_\ll, \text{med}, \text{v\_big}), 1, 1\rangle\}$ and $H_2 = \{\langle \text{str}(O_\ll, \text{med}, \text{v\_high}), 1, 1\rangle, \langle \text{str}(O_\ll, \text{med}, \text{v\_large}), 1, 1\rangle\}$ and hence, the algorithm returns them as incomparable. When comparing P_1 = h_med $\oplus$ med with P_2 = med $\oplus$ l_med, $H_1 = H_2 = \emptyset$ and $E = \{\langle \text{str}(O_\ll, \text{med}, \text{med}), 0, 2\rangle\}$. Therefore, COMPARE-WM($E, P_1, P_2, O_<, '='$) is called where

$L_1 = \{\langle \text{str}(O_<, \text{l\_med}, \text{h\_med}), 2, 1\rangle, \langle \text{str}(O_<, \text{l\_med}, \text{h\_med}), 1, 1\rangle\}$
$L_2 = \{\langle \text{str}(O_<, \text{l\_med}, \text{h\_med}), 1, 1\rangle, \langle \text{str}(O_<, \text{l\_med}, \text{h\_med}), 0, 1\rangle\}$

Because, $\geq_{\text{value}} (L_1, L_2)$ holds, the procedure returns $'\succ'$. As O_1 and O_2 are now empty, $'\succ'$ becomes the result of the algorithm, meaning $P_1 \succ P_2$.

Solution techniques

Algorithm 2 is somewhat similar to that presented in (Mittal, S. & Falkenhainer, B. 1990), but it implements a best first search (BFS) by means of a priority queue O of nodes n. For each node n, a set $X_u(n)$ of remaining active but unassigned attributes is maintained. At each iteration, a node n is taken from O, and the assignments of the first attribute $x \in X_u(n)$ are processed. For every assignment $x : d$ that is consistent with the solution of the current node n (i.e. solution(n) $\cup \{x : d\}, C \nvdash \perp$), a new child node is created. If $X_u(n)$ is empty, the activity constraints are fired in order to find a new set of active but unassigned attributes. That is,

$$X_u(n) = \{x_i \mid \text{solution}(n), \mathbf{A} \vdash \text{active}(x_i)\} - X_a(n)$$

where $X_a(n)$ represents the active, but already assigned attributes in node n.

Algorithm 2: SOLVE($\mathbf{X}, \mathbf{D}, \mathbf{C}, \mathbf{A}, P$)

$X_a(n) \leftarrow \{x_i \mid \{\}, \mathbf{A} \vdash \text{active}(x_i)\}$; $n \leftarrow$ createNode(nil, $X_a(n)$);
$O \leftarrow$ createOrderedQueue(); $CP(n) \leftarrow 0$;
$PP(n) \leftarrow \oplus_{x \in \mathbf{X}} \max_{d \in D(x)} P(x : d)$;
PROCESS(first($X_a(n)$), $X_a(n), n, C, A, P, O$);
while $O \neq \emptyset$
do
- $n \leftarrow$ dequeue(O);
- **if** $X_u(n) \neq \emptyset$
  - **then**
    - $x \leftarrow$ first($X_u(n)$);
    - PROCESS($x, n, \mathbf{C}, \mathbf{A}, P, O$);
  - **else**
    - $X_u(n) \leftarrow \{x_i \mid \text{solution}(n), \mathbf{A} \vdash \text{active}(x_i)\} - X_a(n)$;
    - **if** $X_u(n) = \emptyset$
      - **then**
        - $n_{\text{next}} \leftarrow$ first(O);
        - **if** $CP(n) \not\prec PP(n_{\text{first}})$
          - **then return** ($S(n)$);
          - **else**
            - $PP(n) \leftarrow CP(n)$;
            - enqueue($O, n, CP(n), PP(n)$);
      - **else**
        - $x \leftarrow$ first($X_a(n)$);
        - PROCESS($x, n, \mathbf{C}, \mathbf{A}, P, O$);

procedure PROCESS($x, n_{\text{parent}}, \mathbf{C}, \mathbf{A}, P, O$)
for $d \in D(x)$
do
- **if** solution(n_{parent}) $\cup \{x : d\}, \mathbf{C} \nvdash \bot$
  - **then**
    - $n_{\text{child}} \leftarrow$ new node;
    - solution(n_{child}) $\leftarrow$ solution(n_{parent}) $\cup \{x : d\}$;
    - $X_d \leftarrow$ deactivated(solution(n_{child}), $X(n_{\text{parent}})$);
    - $X_{nd}(n_{\text{child}}) \leftarrow X_{nd}(n_{\text{parent}}) - \{x\} - X_d$;
    - $X_a(n_{\text{child}}) \leftarrow X_a(n_{\text{parent}}) \cup \{x\}$; $X_u(n_{\text{child}}) \leftarrow X_u(n_{\text{parent}}) - \{x\}$;
    - $CP(n_{\text{child}}) \leftarrow CP(n_{\text{parent}}) \oplus P(x : d)$;
    - $PP(n_{\text{child}}) \leftarrow CP(n_{\text{child}}) \oplus (\oplus_{x \in X_{nd}(n)} \max_{d \in D(x)} P(x : d))$;
    - enqueue($O, n_{\text{child}}, PP(n_{\text{child}}), CP(n_{\text{child}})$);

In the priority queue O, nodes are maintained by means of two heuristics: committed preference $CP(n)$ and potential preference $PP(n)$. That is, given a node n,

$$CP(n) = \oplus_{x:d \in \text{solution}(n)} P(x : d)$$
$$PP(n) = CP(n) \oplus (\oplus_{x \in X_{nd}(n)} \max_{d \in D(x)} P(x : d))$$

where $X_{nd}(n)$ is the set of unassigned attributes that can still be activated given the partial assignment solution(n) (the actual implementation employs an assumption-based truth maintenance system (de Kleer, J. 1986) to efficiently determine which attribute's activity can no longer be supported). In other words, $CP(n)$ is the preference associated with the partial attribute-value assignment in node n and $PP(n)$ is $CP(n)$ combined with the highest possible preference assignments taken from all the values of the domains of the attributes in X_{nd}. Thus, $PP(n)$ computes an upper boundary on the preference of a DPCSP solution that includes the partial attribute value assignments corresponding to n.

Theorem 1 SOLVE($\mathbf{X}, \mathbf{D}, \mathbf{C}, \mathbf{A}, P$) *is admissible*
Proof: SOLVE($\mathbf{X}, \mathbf{D}, \mathbf{C}, \mathbf{A}, P$) *is a BFS guided by a heuristic function* $PP(n) = CP(n) \oplus h(n)$, *where* $CP(n)$ *is the actual preference of node n and* $h(n) = \oplus_{x \in X_{nd}(n)} \max_{d \in D(x)} P(x : d)$. *It follows from the previous discussion that* $h(n)$ *is greater than or equal to the combined preference of any value-assignment of unassigned attributes that is consistent with the partial solution of n. In this BFS, the nodes n are maintained in a priority queue in descending order of* $PP(n)$. *Let* δ *be a distance function that reverses the preference ordering such that* $\delta(P_1) \prec \delta(P_2) \leftarrow P_1 \succ P_2$. SOLVE($\mathbf{X}, \mathbf{D}, \mathbf{C}, \mathbf{A}, P$) *can then be described as a BFS guided by* $\delta(PP(n)) = \delta(CP(n)) \oplus \delta(h(n))$, *where the nodes n are maintained in a priority queue in ascending order of* $\delta(PP(n))$ *and where* $\delta(h(n))$ *is a lower bound on the distance between n and the optimal solution. Therefore,* SOLVE($\mathbf{X}, \mathbf{D}, \mathbf{C}, \mathbf{A}, P$) *is an A* algorithm, guaranteed to find a solution S with a minimal* $\delta(P(S))$ *or a maximal* $P(S)$.

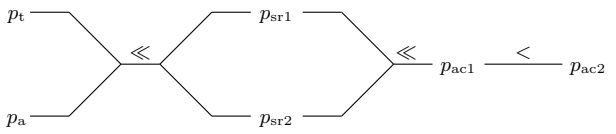

Figure 4: OMP scale for sample configuration task

Applications

This section presents two sample applications of the the DPCSP configuration and compositional modelling.

Configuration

The car configuration problem, originally presented in (Mittal, S. & Falkenhainer, B. 1990), is herein extended with OMPs. The original DCSP specification is as follows:

Attributes and domains		
Attribute	Meaning	Domain
x_p	Package	{luxury,deluxe,standard}
x_f	Frame	{convertible,sedan,hatchBack}
x_e	Engine	{small,medium,large}
x_b	Battery	{small,medium,large}
x_s	Sunroof	{sr1,sr2}
x_a	Airconditioning	{ac1,ac2}
x_o	Opener	{auto,manual}
x_g	Glass	{tinted,non-tinted}

Activity constraints	
$a_{1,2,3}$	active(x_p) $\wedge$ active(x_f) $\wedge$ active(x_e) $\leftarrow \top$
a_4	active(x_s) $\leftarrow$ x_p : luxury
a_5	active(x_a) $\leftarrow$ x_p : luxury
a_6	active(x_s) $\leftarrow$ x_p : deluxe
a_7	active(x_o) $\leftarrow$ x_s : sr2
a_8	active(x_a) $\leftarrow$ x_p : sr1
a_9	active(x_g) $\leftarrow$ active(x_s)
a_{10}	active(x_b) $\leftarrow$ active(x_e)
a_{11}	active(x_s) $\leftarrow$ active(x_o)
a_{12}	active(x_s) $\leftarrow$ active(x_g)
a_{13}	$\neg$active(x_o) $\leftarrow$ x_s : sr1
a_{14}	$\neg$active(x_s) $\leftarrow$ x_f : convertible
a_{15}	$\neg$active(x_a) $\leftarrow$ x_b : small $\wedge$ x_e : small

Compatibility constraints	
c_1	x_p : standard $\wedge$ x_a : ac2 $\rightarrow \bot$
c_2	x_p : luxury $\wedge$ x_a : ac1 $\rightarrow \bot$
c_3	x_p : standard $\wedge \neg(x_f$: convertible$) \rightarrow \bot$
c_4	x_o : auto $\wedge$ x_a : ac1 $\wedge \neg(x_b$: medium$) \rightarrow \bot$
c_5	x_o : auto $\wedge$ x_a : ac2 $\wedge \neg(x_b$: large$) \rightarrow \bot$
c_6	x_s : sr1 $\wedge$ x_a : ac2 $\wedge$ x_g : tinted $\rightarrow \bot$

In this example, the partial preference ordering of figure 4 is assumed. The BPQs $p_{\text{t}}, p_{\text{a}}, p_{\text{sr1}}, p_{\text{sr2}}, p_{\text{ac1}}, p_{\text{ac2}}$ respectively stand for the preference contribution of tinted glass (x_g : tinted), automatic opener (x_o : auto), sr1 sunroof (x_s : sr1), sr2 sunroof (x_s : sr2), ac1 airconditioner (x_a : ac1) and ac2 airconditioner (x_a : ac2). This customer prioritises the choice of airconditioner over the choice of sunroof and the latter over tinted glass and automatic opener. She also prefers the ac2 airconditioner over the ac1, she is indifferent between the sunroof choices and between the tinted glass option and the automatic opener.

A trace of the search is presented below. In order to avoid excessive details in the notation, only the most important actions in running the algorithm are shown, i.e. enqueueing a node in the priority queue, dequeueing the preferred node from the queue, processing the values of an attribute and finding newly

activated attributes when a node without remaining active attributes is encountered.

find active attributes:
enqueue n_0 with $n_0.X_a = \{x_p, x_f, x_e, x_b\}$
dequeue n_0**, process** x_p**:**
enqueue n_1: x_p : luxury with $PP = p_t \oplus p_a \oplus p_{sr1} \oplus p_{ac2}$
enqueue n_2: x_p : deluxe with $PP = p_t \oplus p_a \oplus p_{sr1} \oplus p_{ac2}$
enqueue n_3: x_p : standard with PP = nil
dequeue n_1**, process** x_f**:**
enqueue n_4: x_f : convertible with $PP = p_{ac2}$
enqueue n_5: x_f : sedan with $PP = p_t \oplus p_a \oplus p_{sr1} \oplus p_{ac2}$
enqueue n_6: x_f : hatchBack with $PP = p_t \oplus p_a \oplus p_{sr1} \oplus p_{ac2}$
dequeue n_5**, process** x_e**:**
enqueue n_7: x_e : small with $PP = p_t \oplus p_a \oplus p_{sr1} \oplus p_{ac2}$
enqueue n_8: x_e : medium with $PP = p_t \oplus p_a \oplus p_{sr1} \oplus p_{ac2}$
enqueue n_9: x_e : large with $PP = p_t \oplus p_a \oplus p_{sr1} \oplus p_{ac2}$
dequeue n_7**, process** x_b**:**
ignore x_b : small, it is inconsistent due to a_5 and a_{15}
enqueue n_{10}: x_b : medium with $PP = p_t \oplus p_a \oplus p_{sr1} \oplus p_{ac2}$
enqueue n_{11}: x_b : large with $PP = p_t \oplus p_a \oplus p_{sr1} \oplus p_{ac2}$
dequeue n_{10}**, find active attributes:**
enqueue n_{12}, with $n_{12}.X_a = \{x_s, x_a, x_g\}$
dequeue n_{12}**, process** x_s**:**
enqueue n_{13}: x_s : sr1 with $PP = p_t \oplus p_{sr1} \oplus p_{ac2}$
enqueue n_{14}: x_s : sr2 with $PP = p_t \oplus p_a \oplus p_{sr2} \oplus p_{ac2}$
dequeue n_{13}**, process** x_a**:**
enqueue n_{15}: x_a : ac1 with $PP = p_t \oplus p_a \oplus p_{sr2} \oplus p_{ac1}$
enqueue n_{16}: x_a : ac2 with $PP = p_t \oplus p_a \oplus p_{sr2} \oplus p_{ac2}$
dequeue n_{16}**, process** x_g**:**
enqueue n_{17}: x_g : tinted with $PP = p_t \oplus p_a \oplus p_{sr2} \oplus p_{ac2}$
enqueue n_{18}: x_g : non-tinted with $PP = p_a \oplus p_{sr2} \oplus p_{ac2}$
dequeue n_{17}**, find active attributes:**
enqueue n_{19}, with $n_{19}.X_a = \{x_o\}$
dequeue n_{19}**, process** x_0**:**
ignore x_o : auto, it is inconsistent due to c_5
enqueue n_{20}: x_o : manual with $PP = p_t \oplus p_{sr2} \oplus p_{ac2}$

At this point in the trace, constraint c_5 prevented the assignment x_o : auto. The first element in the priority queue O is now n_{11} because it has a higher PP than any other node that has not been dequeued yet. Part of remainder of the trace, from dequeueing n_{11} onwards, is as follows:

dequeue n_{11}**, find active attributes:**
enqueue n_{21}, with $n_{21}.X_a = \{x_s, x_a, x_g\}$
dequeue n_{21}**, process** x_s**:**
enqueue n_{22}: x_s : sr1 with $PP = p_t \oplus p_{sr1} \oplus p_{ac2}$
enqueue n_{23}: x_s : sr2 with $PP = p_t \oplus p_a \oplus p_{sr2} \oplus p_{ac2}$
dequeue n_{23}**, process** x_a**:**
enqueue n_{24}: x_a : ac1 with $PP = p_t \oplus p_a \oplus p_{sr2} \oplus p_{ac1}$
enqueue n_{25}: x_a : ac2 with $PP = p_t \oplus p_a \oplus p_{sr2} \oplus p_{ac2}$
dequeue n_{25}**, process** x_g**:**
enqueue n_{26}: x_g : tinted with $PP = p_t \oplus p_a \oplus p_{sr2} \oplus p_{ac2}$
enqueue n_{27}: x_g : non-tinted with $PP = p_a \oplus p_{sr2} \oplus p_{ac2}$
dequeue n_{27}**, find active attributes:**
enqueue n_{28}, with $n_{28}.X_a = \{x_o\}$
dequeue n_{28}**, process** x_o**:**
enqueue n_{29}: x_o : auto with $PP = p_t \oplus p_a \oplus p_{sr2} \oplus p_{ac2}$
enqueue n_{30}: x_o : manual with $PP = p_t \oplus p_{sr2} \oplus p_{ac2}$
dequeue n_{29}**, solution found:**
x_p : luxury, x_f : sedan, x_e : small, x_b : large,
x_s : sr2, x_a : ac2, x_g : tinted, x_o : auto

Finally, node n_{29} contains a solution because no further assignments are possible and there is no open node n_i such that $PP(n_i \succ PP(n_{29})$. Other solutions can be found by proceeding with the search after n_{29} has been dequeued, until $\{n_j \in O \mid PP(n_j) \not\prec PP(n_{29})\} \neq \emptyset$.

Compositional modelling

The following DPCSP has been generated from a compositional modelling task in population dynamics (adapted from (Keppens, J. & Shen, Q. 2000)). It is based on a scenario consisting of three species 'prey_1', 'prey_2' and 'pred', where pred feeds on 'prey_1' and 'prey_2', and where 'prey_1' and 'prey_2' compete with each other over scarce resources. The DCSP specification of the problem is as follows:

Attributes and domains

Attribute	Meaning	Domain
x_1	relevant growth(prey_1)	{yes,no}
x_2	relevant growth(prey_2)	{yes,no}
x_3	relevant growth(pred)	{yes,no}
x_4	relevant pred(pred,prey_1)	{yes,no}
x_5	relevant pred(pred,prey_2)	{yes,no}
x_6	relevant comp(prey_1,prey_2)	{yes,no}
x_7	model growth(prey_1)	{exp,log,oth}
x_8	model growth(prey_2)	{exp,log,oth}
x_9	model growth(pred)	{exp,log,oth}
x_{10}	model pred(pred,prey_1)	{hol,lv}
x_{11}	model pred(pred,prey_2)	{hol,lv}

Activity constraints

$a_{1,2,3}$	$\text{active}(x_1) \wedge \text{active}(x_2) \wedge \text{active}(x_3) \leftarrow \top$
a_4	$\text{active}(x_4) \leftarrow x_1 : \text{yes} \wedge x_3 : \text{yes}$
a_5	$\text{active}(x_5) \leftarrow x_2 : \text{yes} \wedge x_3 : \text{yes}$
a_6	$\text{active}(x_6) \leftarrow x_1 : \text{yes} \wedge x_2 : \text{yes}$
a_7	$\text{active}(x_7) \leftarrow x_1 : \text{yes}$
a_8	$\text{active}(x_8) \leftarrow x_2 : \text{yes}$
a_9	$\text{active}(x_9) \leftarrow x_3 : \text{yes}$
a_{10}	$\text{active}(x_{10}) \leftarrow x_4 : \text{yes}$
a_{11}	$\text{active}(x_{11}) \leftarrow x_5 : \text{yes}$

Compatibility constraints

c_1	$x_6 : \text{yes} \wedge (x_7 : \text{exp} \vee x_7 : \text{oth}) \rightarrow \bot$
c_2	$x_6 : \text{yes} \wedge (x_8 : \text{exp} \vee x_8 : \text{oth}) \rightarrow \bot$
c_3	$x_{10} : \text{hol} \wedge (x_9 : \text{exp} \vee x_9 : \text{oth}) \rightarrow \bot$
c_4	$x_{10} : \text{hol} \wedge x_7 : \text{oth} \rightarrow \bot$
c_5	$x_{11} : \text{hol} \wedge (x_9 : \text{exp} \vee x_9 : \text{oth}) \rightarrow \bot$
c_6	$x_{10} : \text{hol} \wedge x_8 : \text{oth} \rightarrow \bot$
c_7	$x_{10} : \text{lv} \wedge (x_9 : \text{exp} \vee x_9 : \text{log}) \rightarrow \bot$
c_8	$x_{10} : \text{lv} \wedge (x_7 : \text{exp} \vee x_7 : \text{log}) \rightarrow \bot$
c_9	$x_{11} : \text{lv} \wedge (x_9 : \text{exp} \vee x_9 : \text{log}) \rightarrow \bot$
c_{10}	$x_{11} : \text{lv} \wedge (x_8 : \text{exp} \vee x_8 : \text{log}) \rightarrow \bot$

The compositional modeller employs two types of assumption class. Attributes x_1 to x_6 corresponds decisions on the relevance of particular phenomena in the model. The assumptions in these classes represent the relevance of the population growth phenomena of the 3 species (x_1, x_2 and x_3), of the predation phenomena (x_4 and x_5) and of the competition phenomenon (x_6). Attributes x_6 to x_{11} corresponds to alternative models for these phenomena. Three types of population growth model are considered: exponential ('exp'), logistic ('log') and other ('oth'). Two types of predation model are considered: Lotka-Volterra ('lv') and Holling ('hol').

The activity constraints are fairly straightforward to understand. In essence, they state that all models for the predation and competition phenomena rely on the existence of population growth models for the species that are involved. The compatibility constraints restrict the combinations of alternative partial models. These were automatically derived from model fragments given in (Keppens, J. & Shen, Q. 2000). For example, c_3 states that the Holling predation model for 'pred' and 'prey_1' can not be combined with the exponential or other growth model of 'pred'.

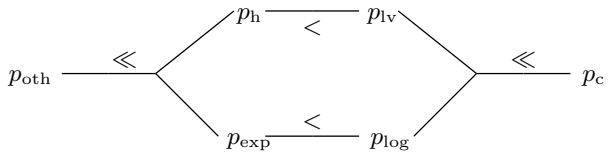

Figure 5: Sample OMP scale for population dynamics

Many models meet the criteria imposed by the constraints. However, the user of the model typically has certain preferences for the different modelling choices that can be made. Figure 5 shows a sample ordering of preferences expressed in the OMP calculus described earlier. In this example, a preference scale is presumed where $\mathbb{B} = \{p_{\mathrm{c}}, p_{\mathrm{lv}}, p_{\mathrm{hol}}, p_{\mathrm{log}}, p_{\mathrm{exp}}, p_{\mathrm{oth}}\}$, respectively representing the utility derived from a model of competition, Lotka-Volterra predation, Holling predation, logistic growth, exponential growth and other growth. The ordering of BPQs in figure 5 states that the contribution of a competition model is an order of magnitude greater than other partial models and 'other population growth' is an order of magnitude lower. Also, Lotka-Volterra predation is deemed more appropriate than Holling predation and logistic growth more appropriate than exponential growth.

Due to space constraints, the trace for this algorithm is not shown. The optimal solution is $x_1 : \mathrm{yes}, x_2 : \mathrm{yes}, x_3 : \mathrm{yes}, x_4 : \mathrm{yes}, x_5 : \mathrm{yes}, x_6 : \mathrm{yes}, x_7 : \mathrm{log}, x_8 : \mathrm{log}, x_9 : \mathrm{log}, x_{10} : \mathrm{hol}, x_{11} : \mathrm{hol}$, with a total preference of $p_{\mathrm{c}} \oplus 2 \times p_{\mathrm{h}} \oplus 3 \times p_{\mathrm{log}}$

Conclusions and Future Work

This paper has introduced a preference calculus based on order of magnitude reasoning. This calculus produces a partial ordering of valuations and distinguishes between different degrees of coarseness. Thus, it better suits the expression of user preferences than approaches employing preferences that are totally ordered. It is most useful in situations where formal preference elicitation is impractical or particularly difficult.

The preference calculus is intended to be used to help solve synthesis problems, such as compositional modelling and configuration. It is used as part of a novel type of constraint satisfaction problem (CSP), the dynamic preference CSP (DPCSP) that incorporates features from dynamic *and* valued CSPs. From dynamic CSPs, a DPCSP takes activity constraints that govern which attributes and the corresponding constraints are part of the problem to be solved. From valued CSPs, a DPCSP borrows the concept of assigning preference valuations to domain values which can be combined to compute the overall preference contribution of an emerging solution. A solution algorithm for such DPCSPs has been presented and its usage was demonstrated by applying it to a DPCSP generated for a compositional modelling task and another for a configuration problem.

Future work includes the development of alternative solution techniques for DPCSPs. One source of inefficiency of the algorithms presented herein is their insistence on finding an optimal solution. However, for many practical applications, finding a close to optimal solution often suffices (Tsang, E. & Warwick, T. 1990). A future focus of this research is therefore to investigate the use of genetic algorithms for solving a DPCSP, which would allow such flexible optimisation.

Acknowledgements

The first author was supported by a scholarship of the Faculty of Science and Engineering of the University of Edinburgh. Both authors are grateful to Ian Miguel and Joe Halliwell for useful discussions.

References

Birmingham, W.P.; Gupta, A.P.; and Siewiorek, D.P. 1992. *Automating the Design of Computer Systems*. Jones and Bartlett.

Bistarelli, S.; Montanari, U.; and Rossi, F. 1997. Semiring-based constraint satisfaction and optimization. *Journal of the ACM* 44(2):201–236.

Boutilier, C.; Brafman, R.; Geib, C.; and Poole, D. 1997. A constraint-based approach to preference elicitation and decision making. In Doyle, J., and Thomason, R.H., eds., *Working Papers of the AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, 19–28.

Brewka, G.; Benferhat, S.; and Le Berre, D. 2002. Qualitative choice theory. In *to appear in Proceedings of the 8th International Conference on Knowledge Representation and Reasoning*.

de Kleer, J. 1986. An assumption-based TMS. *Artificial Intelligence* 28:127–162.

Delgrande, J.P., and Schaub, T. 2000. Expressing preferences in default logic. *Artificial Intelligence* 123(1–2):41–87.

Doyle, J., and Thomason, H. 1999. Background to qualitative decision theory. *AI Magazine* 20(2):55–68.

Falkenhainer, B., and Forbus, K.D. 1991. Compositional modeling: finding the right model for the job. *Artificial Intelligence* 51:95–143.

Ha, V., and Haddawy, P. 1997. Problem-focused incremental elicitation of multi-attribute utility models. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, 215–222.

Keppens, J., and Shen, Q. 2000. Towards compositional modelling of ecological systems via dynamic flexible constraint satisfaction. In *Proceedings of the 14th International Workshop on Qualitative Reasoning about Physical Systems*, 74–82.

Keppens, J., and Shen, Q. 2001. On compositional modelling. *Knowledge Engineering Review* 16(2):157–200.

Levy, A.Y.; Iwasaki, Y.; and Fikes, R. 1997. Automated model selection for simulation based on relevance reasoning. *Artificial Intelligence* 96:351–394.

Mittal, S., and Falkenhainer, B. 1990. Dynamic constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, 25–32.

Mittal, S., and Frayman, F. 1989. Towards a generic model of configuration tasks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 2, 1395–1401.

Nayak, P.P., and Joskowicz, L. 1996. Efficient compositional modeling for generating causal explanations. *Artificial Intelligence* 83:193–227.

Raiman, O. 1991. Order of magnitude reasoning. *Artificial Intelligence* 51:11–38.

Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 631–637.

Tsang, E., and Warwick, T. 1990. Applying genetic algorithms to constraint satisfaction optimization problems. In *Proceedings of the 9th European Conference on Artificial Intelligence*, 649–654.

Tversky, A., and Thaler, R.H. 1990. Anomalies preference reversal. *Journal of Economic Perspectives* 4:201–211.