CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Lexicographically-ordered constraint satisfaction problems
Authors	Freuder, Eugene C.;Heffernan, Robert;Wallace, Richard J.;Wilson, Nic
Publication date	2010
Original Citation	Freuder, EC; Heffernan, R; Wallace, RJ; Wilson, N; (2010) 'Lexicographically-ordered constraint satisfaction problems'. Constraints, 15 (1):1-28. doi: 10.1007/s10601-009-9069-0
Type of publication	Article (peer-reviewed)
Link to publisher's version	http://link.springer.com/article/10.1007%2Fs10601-009-9069-0 - 10.1007/s10601-009-9069-0
Rights	© Springer Science + Business Media, LLC 2009. The final publication is available at http://link.springer.com/article/10.1007%2Fs10601-009-9069-0
Download date	2026-10-31 06:23:27
Item downloaded from	https://hdl.handle.net/10468/1105



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

*Manuscript
Click here to download Manuscript: wallace-con43-revision2.ps.ps
Click here to view linked References

Lexicographically-Ordered Constraint Satisfaction Problems *

Eugene C. Freuder, Robert Heffernan, Richard J. Wallace, Nic Wilson

Cork Constraint Computation Center and Department of Computer Science
University College Cork, Cork, Ireland
email: {e.freuder,r.heffernan,r.wallace,n.wilson}@4c.ucc.ie

Abstract. We describe a simple CSP formalism for handling multi-attribute preference problems with hard constraints, one that combines hard constraints and preferences so the two are easily distinguished conceptually and for purposes of problem solving. Preferences are represented as a lexicographic order over complete assignments based on variable importance and rankings of values in each domain. Feasibility constraints are treated in the usual manner. Since the preference representation is ordinal in character, these problems can be solved with algorithms that do not require evaluations to be represented explicitly. This includes ordinary CSP algorithms, although these cannot stop searching until all solutions have been checked, with the important exception of heuristics that follow the preference order (lexical variable and value ordering). We describe relations between lexicographic CSPs and more general soft constraint formalisms and show how a full lexicographic ordering can be expressed in the latter. We discuss relations with (T)CP-nets, highlighting the advantages of the present formulation, and we discuss the use of lexicographic ordering in multiobjective optimisation. We also consider strengths and limitations of this form of representation with respect to expressiveness and usability. We then show how the simple structure of lexicographic CSPs can support specialised algorithms: a branch and bound algorithm with an implicit cost function, and an iterative algorithm that obtains optimal values for successive variables in the importance ordering, both of which can be combined with appropriate variable ordering heuristics to improve performance. We show experimentally that with these procedures a variety of problems can be solved efficiently, including some for which the basic lexically ordered search is infeasible in practice.

1 Introduction

An important challenge for constraint solving is to incorporate user preferences into the problem representation so that solutions can satisfy these preferences as well as hard constraints. This is necessary if constraint technology is to be used to tackle standard decision making problems with multiple objectives and attributes.

The constraint satisfaction paradigm appears well-suited for representing and solving problems composed of a mixture of 'hard' feasibility constraints and 'soft' con-

* This work received support from Science Foundation Ireland under Grants 00/PI.1/C075 and 05/IN/1886.

straints, including those that represent user preferences. Such problems are often handled by formulations in which constraint elements are associated with evaluations that allow comparisons between these elements [1, 2]. In the present work, we use a special soft-constraint strategy in which a preference ordering is imposed on complete assignments, in terms of variables and their assigned values. This ordering is *lexicographic* in form, which means that a good assignment for a more-preferred variable is more important than a good assignment for a less-preferred variable in deciding the overall ranking of solutions. The preference ordering is assumed to be independent of any constraints that may hold among these variables. The constraints, therefore, restrict the alternatives given by an ideal preference ordering to those that can actually be realized.

This form of soft constraint system is a special case of the "lexicographic CSP" or "lex-VCSP" defined by [3]. As these authors show, lex-VCSPs are in turn equivalent to a kind of weighted CSP (cf. Section 3). However, because of the character of the ordering in our case, we do not need to represent preferences numerically, and we can build up partial solutions correctly without reference to numerical operations such as addition. In fact, this form of representation is in the spirit of the "ordinalist" view of utility, (i.e. the interpretation of utility functions as representing the quantitative structure of (ordinal) preference relations) [4], as well as qualitative approaches to representing preferences that have emerged in AI [5]. So, while we will follow [3] and refer to the present representation as a "lexicographic CSP", it is a very special case of the class that they describe, with implications both for its usefulness as a representation in the context of preferences and its ability to support efficient algorithms. For this reason, we will use the term "lex-VCSP" to refer to the more general category of CSPs whose evaluations can be ordered lexicographically.

Lexicographic CSPs (in the present sense) are potentially useful in applications that involve multiple objectives and attributes, where attribute values comprise small finite sets, and where feasibility constraints impose restrictions on assignments that are actually possible. We are particularly concerned with problems in which attributes are either qualitative in nature or take on values from a small discrete set. In these cases, an ordinal model requiring stringent but reasonable assumptions may be an effective decision aid.

For example, consider a customer who has the overall objective of buying a good-quality digital camera. This might entail more specific objectives that can be expressed as: "I prefer any camera with 2 megapixels over any camera with 1 megapixel, but if they both have the same number of megapixels, I'll choose the lighter one". In other words, pixels has priority over weight, with a larger value preferred in the first case and a smaller in the second. In addition, let us suppose that a digital zoom is preferred to an optical zoom, and this attribute has a priority between pixels and weight. To round out this example, we will introduce some feasibility constraints: we suppose that a larger number of pixels is associated with greater weight, and that there are also restrictions between pixels and zoom-type and zoom-type and weight.

We represent these user requirements using a well-known technique from applied decision analysis, and then show how the same example is represented as a lexicographic CSP. The decision analysis involves constructing a hierarchy of objectives, starting with a general and rather ill-defined objective that one desires, and dividing or

specializing it into more specific objectives, which are then associated with measurable attributes. (In the most common forms of decision analysis, numerical preference functions would then be constructed that take into account priorities and tradeoffs among these attributes [6, 7].)

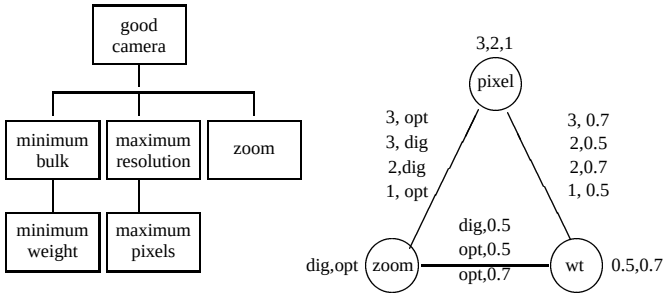


Fig. 1. Multiattribute decision problem, represented by a hierarchy of objectives on the left with measurable attributes at the lowest level and by a lexicographic CSP on the right, which incorporates tradeoffs and restrictions as hard constraints. (Constraints are shown as viable tuples.) In this example, the importance ordering over variables is pixel $>$ zoom $>$ weight; also, the values in each domain are ordered from left to right.

Putting this problem in the form of an objectives hierarchy gives the result shown on the left in Figure 1. Both user requirements and tradeoffs due to feasibility constraints can be represented by the lexicographic CSP on the right in Figure 1. An important benefit that is immediately evident from this example is that with lexicographic CSPs there is a clean separation between preferences and feasibility constraints, so the two can be incorporated into one system in a way that does not obscure either of them. Another potential benefit is the ease of merging standard preference elicitation techniques with this constraint representation.

The lexicographic CSP as defined here is amenable to certain extensions, each of which enhances the usefulness of this form of representation. In the first place, such an ordering might apply to only a subset of the CSP variables. We call this an "embedded preference ordering". In the second place, as Brewka and others have shown, it is sometimes useful to consider the set of lexicographic orderings that are consistent with a given partial order, which allows more decisive comparisons between alternatives [8, 9]. More generally, it may be useful to consider different importance orderings and their associated solutions [10], and this can be done if we have efficient algorithms.

Some recent approaches to qualitative representation of preferences have focused on "conditional preferences", where the preference ordering with respect to a given attribute is dependent on selection of a value for a different attribute [11]. (Suppose, for example, that our customer prefers the digital to the optical zoom with a 3 megapixel camera, but prefers optical to digital zoom with a 1 megapixel camera.) As it turns out, conditional preferences can also be incorporated into the lexicographic CSP representation; this is especially important because for lexicographic CSPs, comparing alterna-

tives with respect to preference is always easy, in contrast to CP-nets. In addition, there are important relations between lexicographic orderings and the *ceteris paribus* orderings of "CP-net" representations. In particular, it has been shown that acyclic CP-net orderings are dominated by conditional lexicographic orderings [12, 13].

A distinctive feature of lexicographic CSPs is that these problems can be solved using standard algorithms for ordinary CSPs. This contrasts with the general case for soft constraint systems, where branch and bound techniques must be used to obtain provably optimal solutions. In addition, the structure of lexicographic CSPs allows efficient algorithms to be devised that are specialized for this type of problem.

Junker has emphasized that in qualitative preference systems as opposed to multi-criteria optimisation problems (which use globalized optimisation criteria), preferences can be used to support specific heuristic decisions in order to make search more efficient [14, 9]. This paper demonstrates that this is particularly true for lexicographic CSPs.

Although lexicographic orderings have a venerable history in the study of preference in such areas as economics and decision making [15], there has been relatively little work on applying this idea to CSPs, and on developing algorithms to solve these problems. [16] introduced the idea of a lexicographic ordering on sets of constraint valuations, an idea developed further in [3], as already noted. [14] considered multi-criteria preferences in connection with constrained optimization, and lexical orderings were used in some cases to compare solutions (called "B-preferences"). Work on constraint hierarchies [17] can also be cited, since comparisons between assignments depend on the lexical ordering of the levels of the hierarchy, so that a constraint violation at level k overrides any number of violations at higher levels. [18] have studied consistency (GAC) algorithms for constraints that induce lexicographic orderings on paired vectors of variables. As our introductory example suggests, combining lexicographic orderings with constraint-based reasoning may give a useful representation for many decision problems that is also amenable to the powerful algorithms and heuristics developed in connection with CSPs.

The remainder of the paper is organized as follows. Section 2 gives a formal definition of lexicographic CSPs. Section 3 discusses relations to other formal representations that can be applied to the the kinds of problems we are interested in. Section 4 discusses strengths and limitations of a lexicographically based representation of constrained preferences. Sections 5 and 6 discuss search strategies for lexicographic CSPs, including experimental comparisons on test problems. Section 5 describes search procedures based on ordinary CSP algorithms. Section 6 describes a branch and bound and a specialized lexical search algorithm. Section 7 summarises the main conclusions.

This paper incorporates and extends some of the results presented in [19].

2 Definition of Lexicographic CSP

Definition 1. Lexicographic CSP. A finite CSP is defined in the usual way as a triple $\langle V, D, C \rangle$, where V is a set of variables, D is a set of domains each of which is associated with a member of V, and C is a set of constraints, or relations holding between subsets of variables.

To specify a CSP as lexicographic, we introduce the following definitions. A labelling of set V is a bijection between $\{1, \ldots, |V|\}$ and V. A *lexicographic structure* L over V is a pair $\langle \lambda, \{>_X\colon X \in V\}\rangle$, where the second component is a family of total orders, with $>_X$ being a total order on the domain of X, and λ is a labelling of V. We write the labelling λ of V as $X_1, \ldots, X_n$ so that for each i, $X_i = \lambda(i)$, and $n = |V|$. The associated *lexicographic order* $>_L$ on (complete) assignments is defined as follows: $\alpha >_L \beta$ if and only if $\alpha \neq \beta$ and $\alpha(X_i) >_{X_i} \beta(X_i)$, where X_i is the first variable (i.e., with minimum i) such that α and β differ.

A *lexicographic CSP* is a tuple $\langle V, D, C, \lambda, \{>_X\colon X \in V\}\rangle$, where $\langle V, D, C\rangle$ is a finite CSP and $\langle \lambda, \{>_X\colon X \in V\}\rangle$ is a lexicographic structure over V.

A solution to a lexicographic CSP is an assignment α^* such that

(i) α^* is a satisfying assignment, that is, it is consistent with, or satisfies, all constraints in C.
(ii) $\alpha^* >_L \alpha$ holds for any other satisfying assignment α.

3 Comparisons with Other Formulations

3.1 Lexicographic CSPs and soft constraint systems

In constraint-based reasoning, preferences are often modeled as soft constraints, in which failure to satisfy a constraint serves to deprecate the offending values, but does not lead to outright exclusion. The most important formalisms are the valued and semiring CSPs, in which evaluations are associated with domain values and with either constraints or constraint tuples [1, 3]. Under the proper assumptions (especially preferential independence and scale equivalence), these evaluations can be used to represent preferences (in the form of utilities) as well as other scalable features like importance or likelihood. The resulting problems are constraint optimization problems, in which solutions are sought that optimize some function of the evaluations, for example the minimum sum or the largest minimum value associated with any value or tuple in an assignment.

Each soft constraint framework includes several classes that are distinguished by the operators used to combine and compare evaluations. Of greatest relevance here are the fuzzy CSPs and the weighted CSPs. The former uses the max operator to combine evaluations and selects the minimum maximum evaluation associated with a violated constraint. The latter combines by summing evaluations and selects the minimum sum. Lexicographic CSPs can be classified in these terms as a kind of weighted CSP. This can be shown by embedding a lexicographic ordering within the weighted CSP framework as follows:

Lexicographic CSP as a weighted CSP. For each $i = 1, \ldots, n$ we define a unary weighted constraint W_i over variable X_i, given by $W_i(x) = kb^{n-i}$, where x is the kth best value in the domain of X_i and b is the largest domain size. Then for assignments α and β, the sum of the weights associated to α is less than the sum associated to β if and only if $\alpha >_L \beta$.

Within this field of research, some previous work has concerned itself specifically with lexicographic orderings. An evaluation structure for CSPs involving a lexicographic ordering was originally developed within the fuzzy CSP context, in order to avoid the limited discriminability between solution values in the normal fuzzy system due to the use of fuzzy min and max operations for combining and comparing evaluations [16]. In this formulation, preferences for k-tuples associated with a given constraint are ordered by increasing magnitude, and two solutions are compared beginning with the first members of each ordering and proceeding through the lists until a difference is found. In addition, constraint priorities are incorporated into this model by associating a priority level with each constraint, and making the evaluation for a tuple the maximum of its preference value and the complement of the priority value, i.e.

$$\mu_S(u_1, \ldots, u_k) \;=\; \max(1 - \alpha_C, \mu_R(u_1, \ldots, u_k))$$

where μ_S and μ_R are evaluations of the fuzzy relations associated with constraint C, and α_C is C's priority level. However, lexicographic CSPs as we define them do not fall under this 'extended' fuzzy model, in which priorities and preferences are balanced.

A more general formulation falling within the valued CSP framework is the lex-VCSP model, where evaluations (associated with constraint violations) are treated as multi-sets and the combinator is multiset-union. An additional top ($\top$) value acts as an absorbing element and can be used to represent violations of hard constraints. Comparison involves sorting the multisets associated with each solution by descending value and comparing them lexicographically, beginning with the highest value and choosing the evaluation with the smaller value for the first difference found [3]. These authors also show that lex-VCSPs are equivalent to weighted CSPs, with positive ∞ serving as the top value. Lexicographic CSPs (in our sense) can, therefore, be subsumed under the lex-VCSP formulation; however, a different form of embedding is required than for weighted CSPs.

Lexicographic CSP as a lex-VCSP. For each $i = 1, \ldots, n$ and each $x \in D(X_i)$, we define a unary soft constraint with associated constraint $X_i \neq x$ and evaluation $t_i(x) = \frac{(n-i)b+k}{nb}$, where x is the kth best value in the domain of X_i and b is the largest domain size. In a lex-VCSP, each complete assignment α is associated with the multiset of all evaluations of soft constraints which α violates, that is $\{t_i(\alpha(X_i)) : i = 1, \ldots, n\}$. The lexicographic comparison of two multisets is performed by first comparing the largest element of each set. In this case, α is preferred to β if and only if the multiset associated with α is lexicographically prior to the multiset associated with β, as described in the previous paragraph. Note that under this construction, if $i < j$ then $t_i(x) > t_j(y)$ for all values x and y. Also, $t_i(x) > t_i(y)$ if and only if x is less-preferred than y. So α is preferred to β if and only if $\alpha(X_i)$ is a better value than $\beta(X_i)$, where i is minimal such that $\alpha(X_i)$ and $\beta(X_i)$ differ. This holds if and only if $\alpha >_L \beta$ according to the lexicographic ordering specified in Definition 1.

The procedure involved here can be clarified by a simple example. Suppose we have a lexicographic CSP with three variables and three values. We use the integers 1,2,3 to represent each domain, with the ordering $1 > 2 > 3$. Similarly, variable 1 is

the most important, variable 3 the least. Then, by the expression just given (split into three components after multiplying out the terms in the numerator), the evaluations associated with successive values are:

values:	1	2	3
var1:	$1 - 1/3 + 1/9$	$1 - 1/3 + 2/9$	$1 - 1/3 + 3/9$
var2:	$1 - 2/3 + 1/9$	$1 - 2/3 + 2/9$	$1 - 2/3 + 3/9$
var3:	$1 - 3/3 + 1/9$	$1 - 3/3 + 2/9$	$1 - 3/3 + 3/9$

Or, with simplified expressions:

$7/9$	$8/9$	1
$4/9$	$5/9$	$6/9$
$1/9$	$2/9$	$3/9$

As usual with weighted-constraint formulations, the objective is to minimize the overall valuation using the combinator described earlier. (To simplify we ignore hard constraints, whose violation would override any of these evaluations.) Thus, in forming multisets of evaluations, any feasible assignment in which variable 1 has the value 1 will have a lower maximum value (7/9) than any assignment in which this variable has value 2 or 3, and this comparison will dominate any comparisons of evaluations related to assignments to variables 2 and 3 (all $\leq$ 6/9). And so forth.

Lexicographic CSPs can also be associated with the constraint hierarchy framework of [17]. In an embedding of this sort, the feasibility constraints are associated with level H_0, the level whose constraints must be satisfied in any feasible solution, and the importance ordering is represented by the subsequent levels in the hierarchy. In this case (unlike most constraint hierarchies), the evaluations of feasible solutions collectively form a total order.

Lexicographic CSP as a constraint hierarchy We define n+1 levels in the hierarchy. H_0 contains all the feasible constraints. For $k = 1, \ldots, n$ we define a level H_k of the constraint hierarchy containing a single unary constraint related to assignments to variable X_k in the importance ordering. For each level k, the locally-better comparator (such as $<$) is based on an error function that returns a different error for each assignment to X_k such that an error for value a is less than that for value b if and only if a comes before b in the total order for $D(X_k)$. This meets the definition of a constraint hierarchy valuation: that valuation α is better than valuation β if, for all constraints through some level $k-1$, the error associated with α is equal to that for β, and at level k the error is strictly less for at least one constraint and less than or equal for all the rest [17]. In addition, α is locally-better than β if and only if $\alpha >_L \beta$.

In each of these three cases, embedding a lexicographic CSP within the more general framework involves making evaluations explicit, either in the form of specific scalar values or as a level in a hierarchy. Thus, in the soft constraint frameworks, the original n-ary soft constraint is represented by $n \times d$ evaluations. But in making this transformation, the most significant representational feature of the lexicographic CSP - the explicit representation of a lexicographic order - is lost, or at least obscured. In other words, when making evaluations explicit, the more general formulations also do away with the original ordering, at least with respect to the way in which they carry out combinations and comparisons. (Intriguingly, in the lex-VCSP the original lexicographic ordering is in a sense 'rediscovered' in a lexicographic ordering on multisets of evaluations; but this is still a much more roundabout way of representing the original problem.) In addition, by explicitly representing the lexicographic ordering, the lexicographic CSP formulation avoids the need to explicitly represent the evaluations as separate elements. Nor is this required in order to search for optimal solutions.

The take-home lesson, then, is that when the assignments form a total order in the fashion of Definition 1, we do not have to represent constraint-weights (or a constraint hierarchy) explicitly. This allows a greatly simplified representation of these problems and supports a variety of specialised procedures for solving them, as described in later sections.

3.2 Lexicographic CSPs and CP-nets

CP-nets are a recently proposed formalism for the qualitative representation of preferences among outcomes with multiple attributes [11, 13]. The intention in this case is to represent *conditional* preferences. Nonetheless, it can be compared to the lexical CSP representation at several points. (And, as shown at the end of this section, the lexicographic CSP framework can be extended to handle conditional preferences, thus providing a further point of comparison.)

Like CSPs, CP-net structures are based on assignments of values to variables, or "features". As noted, CP-nets encode conditional dependencies, in which the preference ordering of values in the domain of variable X_i depends on values assigned to other variables, called the "parents" of X_i. These orderings are stored in a "conditional preference table" (CPT) associated with X_i. A more recent variant, the TCP-net [20, 21], includes elaborations to handle relations of importance between the features of user-selections. This corresponds to the ranking of variables in lexicographic CSPs.

A critical feature of CP-nets is that preferences are only defined under "*ceteris paribus*" conditions. If, for example, features A and B each have two values, a_1, a_2 and b_1, b_2, respectively, and we have unconditional preferences $a_1 >_{X_A} a_2$ and $b_1 >_{X_B} b_2$, then we can deduce from *ceteris paribus* assumptions that $a_1b_1 >_N a_2b_1$, $a_2b_1 >_N a_2b_2$, and hence $a_1b_1 >_N a_2b_2$, but we cannot order a_1b_2 and a_2b_1 on this basis. As a result of this feature, preference orders can be established on the basis of "flipping sequences" (as illustrated in the last example). This is still true of TCP-nets, although in some cases adjacent outcomes in a sequence can be separated by a change in two variables rather than one.

An important difference between (T)CP-nets and lexicographic orderings (and also soft constraints formalisms) is that while comparisons are easy for lexicographic order-

ings, since they are based on successive comparisons of indices, they can be extremely hard for (T)CP-nets, since they depend on finding flipping sequences for transforming one alternative into another [22, 23].

Although (T)CP-nets do not encode feasibility constraints directly, the orderings that they represent can be combined with such constraints in much the same way that lexicographic CSPs combine a particular preference ordering with a constraint representation [24]. With (T)CP-nets, the constrained optimisation problem involves finding the undominated feasible outcomes. For an acyclic CP-net [24], (and similarly, for an acyclic TCP-net [21, 25]) one can use a CSP search algorithm to find a single undominated solution, by instantiating variables in an ordering compatible with the parent-child ordering. A "staged lexical" algorithm (see Section 6.2) can also be adapted for this purpose. To find more than one undominated solution can, however, involve computationally complex comparisons between solutions (although this is not always so: see e.g., [26]).

Except in some trivial cases, the order on assignments generated by a CP-net, or by a TCP-net, is never a lexicographic order. Therefore, in contrast to the systems discussed in the previous subsection, lexicographic CSPs cannot be embedded in a CP-net, even when the latter is extended to incorporate feasibility constraints as in [24]. (See [27] for a formalism that extends both (T)CP-nets and lexicographic orders.) The reason for this is that flipping sequences require that consecutive elements in the ordering differ by at most one (CP-nets) or two (TCP-nets) elements. However, consecutive elements in a lexicographic ordering can differ by up to $|V|$ elements. More precisely (see [27]),

Theorem 1. *Let $>_L$ be a lexicographic order (as defined above) on the set of complete assignments, and for all $X \in V$, $|\underline{X}| > 1$. Then (a) if $|V| > 1$, there exists no CP-net N on V with $>_N = >_L$, (b) if $|V| > 2$, there exists no TCP-net M on V with $>_M = >_L$.*

Proof. Consider any α and β which are consecutive in the order, $\alpha \succ_N \beta$, but there does not exist γ with $\alpha \succ_N \gamma \succ_N \beta$. Because $\alpha \succ_N \beta$, there exists some flipping sequence $\alpha = \alpha_1, \ldots, \alpha_l = \beta$ with α_i an improving flip from α_{i+1}. But because α and β are consecutive, $l = 2$ and α is an improving flip from β. By definition of an improving flip, α and β differ on precisely one variable.

The same argument can be used to show that if α and β are consecutive in the order $\succ_N$ when N is a TCP-net, then (using Lemma 5 of [20]), α is an improving (TCP-)flip from β, so, by definition of a TCP-flip, α and β differ on either one or two variables.

Any lexicographic order on assignments contains consecutive elements α and β which differ on all $|V|$ variables (assuming the domain of each variable has at least two elements). For example, if the domain of each variable is the set $\{1, 2\}$ with the usual ordering, then the assignments $(1, 2, 2, \ldots, 2)$ and $(2, 1, 1, \ldots, 1)$ are consecutive assignments in the lexicographic order which differ on every variable. So if $|V| > 2$ then $\succ_N$ is not a lexicographic order if N is either a CP-net or a TCP-net. □

Conditional Lexicographic Orderings. As noted above, the lexicographic representation that we employ can be extended to represent conditional preferences. This subject is treated at greater length in [12] and [28]; here, we briefly summarise the main ideas.

A conditional lexicographic ordering, like a lexicographic ordering, is based on an ordering $X_1, \ldots, X_n$ of the variables, and to compare two assignments we see which is better on the first variable X_i on which they differ. In this case, the ordering of the values of the domain of X_i can be conditional on values of previous variables.

A conditional lexicographic CSP can be defined as follows:

Definition 2. Conditional lexicographic CSP. Define a *conditional lexicographic structure* over V to be a tuple $K = \langle \lambda, G, CPT \rangle$, where λ is a labelling of V, with $\lambda(i)$ being written X_i, G is a directed acyclic graph on V which is compatible with λ, i.e., $(X_i, X_j) \in G$ implies $i < j$. CPT is a function which associates a conditional preference table $CPT(X)$ to each $X \in V$. Each conditional preference table $CPT(X_i)$ associates a total order $>_u^{X_i}$ with each instantiation u of the parents U_i of X_i (with respect to G). The associated *conditional lexicographic order* $\succ_K$ on assignments is defined as follows: $\alpha \succ_K \beta$ if and only if $\alpha \neq \beta$ and $\alpha(X_i) >_u^{X_i} \beta(X_i)$, where X_i is the first variable (i.e., with smallest i) such that $\alpha(X_i) \neq \beta(X_i)$, and $u = \alpha(U_i) = \beta(U_i)$. It is easily seen that $\succ_K$ is a total order on assignments.

Conditional lexicographic structures as defined above have the requirement that the graph G is compatible with the importance ordering of variables, so that the parents U_i of a variable X_i are more important than the variable X_i. One can relax this assumption, however, and consider the more general case where parent-child relations in the conditional preference network do not necessarily conform to importance relations in the variable ordering. This is done by a strategy of indirection that involves a function Q which assigns a number $Q(x|u)$ for every value x of X_i and assignment u to U_i. The conditional preference order is then defined as follows: to compare assignments α and β we find the first X_i where $Q(\alpha(X_i)|\alpha(U_i))$ is not equal to $Q(\beta(X_i)|\beta(U_i))$. If $Q(\alpha(X_i)|\alpha(U_i))$ is less than $Q(\beta(X_i)|\beta(U_i))$, we prefer α to β; else we prefer β to α. Another way of viewing this is that we are converting each assignment $\alpha = (x_1, \ldots, x_n)$ to an n-tuple of numbers $\alpha' = (Q(x_1|u_1), \ldots, Q(x_n|u_n))$, where u_i is the assignment α makes to U_i. The conditional lexicographic order $>_L$ is then just the standard lexicographic order on these n-tuples of numbers: α is preferred to β if and only if α' is lexicographically less than β'. Hence $>_L$ is still a total order.

This lack of restriction regarding parent-child relations means that a user can express priorities and conditions on preferences independently, without having to be concerned with making the relations correspond. Moreover, such problems can still be solved efficiently, using extensions of the algorithms described below [12, 28].

3.3 Lexicographic CSPs and multiobjective optimisation

Multiobjective combinatorial optimisation (MOCO) and multiobjective programming (MOP) are concerned with finding optimal solutions when there are more than one criteria or objectives, expressed as functions (typically real-valued) on subsets of decision variables. The problem can be stated in the following general terms:

$$\min_{x \in \mathcal{X}} f(x) = (f_1(x), \ldots, f_p(x))$$

where $\mathcal{X} \subset \Re^n$ is the feasible set and f is a vector-valued objective $f : \Re^n \longrightarrow \Re^p$, where $f_i : \mathcal{X} \longrightarrow \Re$ are the objectives [29]. In the basic case, the entire Pareto frontier is considered. Recently, this general approach has been carried over to CSPs [30, 31, 9].

In other cases, acceptable aggregation functions can be used, such as a weighted sum, or the objectives can be ordered. One example of the latter is the lexicographic ordering [32]. A feasible solution $\hat{x}$ is lexicographically optimal if there is no $x \in \mathcal{X}$ such that $f(x) <_{\text{lex}} f(\hat{x})$, where $<_{\text{lex}}$ corresponds to $>_L$ in Definition 1. In most cases, solutions are obtained through some form of weighting scheme (cost functions), similar to that described above in connection with weighted CSPs.

In all these cases, preferences are implicit in that smaller values returned by an objective function are generally considered to be better (e.g. lowest weight, lowest cost). However, in order to deal more explicitly with preferences, especially in the form of utility scales, it is necessary to introduce further specifications and to follow certain restrictions. As an obvious instance of the latter, it is not acceptable to combine objectives considered as "utilities" with other functions that are simply physical or monetary objectives. In addition, we cannot simply assume that some weighted additive formula is an additive utility without establishing the necessary independence conditions (e.g. preferential independence) and deriving coefficients (weights) that truly reflect differences of proportion among attributes [6]. (For a concise statement of this issue, see [33], p. 646.)

There are different strategies for introducing explicit preferences into the multiobjective framework. The first involves relegating multiobjective optimisation and multicriteria decision analysis to different stages. In this case, the output of the MOCO stage is a non-dominated set of solutions; this is then the input used by the multi-criteria decision making technique, e.g. MAUT or AHP, to order the solutions. This is the approach advocated by some authorities [32, 29].

A second approach involves transforming each objective into a utility scale or, more generally a preference ordering, and combining these according to the requirements for valid aggregate preferences. This is the approach taken by Junker (see [9, 10]), who in addition considers ordinal structures for preference representation. In his approach, criteria are defined which are functions of subsets of decision variables; unlike MOCO criteria, these define preference orderings. These criteria are then added to a CSP representation in the form of extra constraints. (A third approach is also possible, in which MOP search returns a [usually nondominated] solution that is 'critiqued' by the decision maker, leading to another bout of combinatorial search under the constraints inferred from the critique, and so forth [33]. Since this approach does not assume an explicit preference ordering at the start, it is outside the scope of the present work.)

In the present work, we also follow the second approach. In this case, a CSP representation allows us to use a multiobjective approach with a purely ordinal representation. Here, we consider only feasible solutions (hard constraints), but this is not a basic limitation, since our methods can be extended to cases where a lexicographic representation of preferences is combined with other soft constraints.

Lexicographic CSPs can be viewed as a special case of the approach used by Junker. The main difference is that we have explored the ramifications of this special case more extensively, including relations to other soft constraint systems and the development

and testing of specialised algorithms. In addition, the present exposition distinguishes between multiobjective optimisation and multiattribute preference representations and presents a more systematic account of their differences.

In his examples, Junker expresses his criteria as extra variables. They are, therefore, not only distinguished constraints but distinct variables. This is comparable to a version of the lexicographic CSP in which the lexicographic ordering holds for a subset of the variables (what we call an embedded lexicographic ordering). However, in our case we do not declare extra variables; instead we use a k-ary rather than an n-ary constraint to represent the preference ordering.

4 Representing Preferences with Lexicographic Orders

The previous section served to locate lexicographic CSPs within the space of multi-attribute representations. There are further issues concerning the manner in which preferences are represented, which are of critical importance for evaluating the applicability of this approach. These are discussed in the present section.

4.1 Representational adequacy and expressiveness

Preferences refer to selections of outcomes from sets of alternatives [34]. (Here, we only consider "riskless choice", where uncertainty is not represented explicitly.) We follow the usual convention in artificial intelligence in not distinguishing between overt choices and verbal judgments of preference. In either case preferences can be represented by binary relations with properties such as transitivity and reflexivity that reflect basic rationality assumptions. Our definition of lexicographic structure is consistent with a preference relation on the set of full CSP assignments, and there is an associated order-preserving (utility) function by virtue of this relation being both transitive and complete [35]. More precisely, the properties of a lexicographic ordering given in Definition 1 imply that the ordering on full assignments is transitive and complete; this follows from the fact that distinct assignments differ in at least one value and that for any two distinct assignments α and β, the ordering of these assignments depends solely on some X_i, the variable with the minimum label i on which they differ. Hence, by a well-known theorem (see [35]), there is a real-valued function u such that for any two assignments, α and β:

$$\alpha \ >_L \ \beta \iff u(\alpha) > u(\beta)$$

Moreover, the axiomatic constructions of Fishburn and Plott, Little and Parks show that, given an ordering on vectors of attribute values that is transitive, antisymmetric and complete, together with an independence condition of the following form:

$$(x_1, \ldots, x_{i-1}, a_i, x_{i+1}, \ldots, x_n) \ >_L \ (x_1, x_{i-1}, \ldots, b_i, x_{i+1}, \ldots, x_n)$$

$$\text{iff } (y_1, \ldots, y_{i-1}, a_i, y_{i+1}, \ldots, y_n) \ >_L \ (y_1, \ldots, y_{i-1}, b_i, y_{i+1}, \ldots, y_n),$$

and an axiom of noncompensation between attribute domains with the following form (where $>_{X_i}$ is the projection of $>_L$ to X_i):

$$\text{if } (x_i >_{X_i} y_i \text{ iff } z_i >_{X_i} w_i)$$

$$\text{and } (y_i >_{X_i} x_i \text{ iff } w_i >_{X_i} z_i)$$

$$\text{then } (x >_L y \text{ iff } z >_L w) \text{ and } (y >_L x \text{ iff } w >_L z),$$

then this ordering *must* be lexicographic [36, 37]. [1] [2] This form of ordinal representation, therefore, can be given a sound axiomatic foundation. In this respect, it can be distinguished from most if not all other ordinal representations.

Lexicographic CSPs are associated with an aggregated utility function, since this function is based on subutilities on individual domains. By using a qualitative representation in this context, we circumvent many difficult issues regarding the validity of a preference representation that arise when one associates numerical values with CSP elements and combines them according to standard operators such as addition. In particular, we do not have to worry about whether we meet the assumptions that, (i) the aggregate (derived) scale values are consistent with preferences based on full alternatives, (ii) the scales for distinct attributes are commensurate, so their values can be meaningfully combined. In the latter case, unless one determines coefficients of proportionality by requiring the user to specify equivalent marginal rates of substitution, it is not clear that summed 'preference values' are genuine additive utilities (cf. [6]).

In this connection, it should be noted that in our running example, while two of the attributes are quantitative in nature, neither can be associated with marginal rates of substitution without grossly distorting the character of the actual sets of attribute values, which are finite and small. This would seem to rule out or at least severely constrain approaches based on multi-attribute utility theory (MAUT) in such domains. Other approaches to multi-attribute scaling, such as AHP [7], involve equally stringent assumptions. Similar criticisms of approaches that involve numerical scaling have been made in the literature on multi-criteria decision making, e.g. [38].

On the other hand, this form of qualitative preference representation has some obvious limitations. Lexicographic orders are an extreme case where there are no tradeoffs in preference between the values of any two attributes. Referring back to the introductory example, suppose that a 4 megapixel camera is available, but the weight of this camera is 7, a ten-fold increase over the next-highest value. In this case, our customer might decide that the extra megapixel does not compensate for the added weight of the camera; but this of course violates a lexicographic ordering, where any improvement in number of pixels will outweigh any increase in weight.

Because of such situations, the practical importance of lexicographic orderings has sometimes been questioned. Perhaps the best-known critique along these lines is that of Keeney and Raiffa [6] (but see also [15]). We must admit that such criticisms have

[1] The axiom of noncompensation says that the ordering on complete assignments depends just on which is better on individual attributes, not on how much better or worse they are on each attribute. This prevents cases where, for example, some $w >_{X_j} z$ comparison overrides a $z >_{X_i} w$ comparison, while the corresponding $y >_{X_j} x$ comparison does not override the $x >_{X_i} y$ comparison. In such cases the consequent may not follow from the antecedents.

[2] In the description above, we have substituted our notation for the original notation of [36]. Since the domain and assignment orderings in the present paper have the same properties as $\succ$ and $\succ_i$, respectively, in [36], the theorem on lexicographicity also follows from the axioms in their present form.

merit. Thus, there will be cases, namely those in which tradeoffs of one sort or another are important, where a system based on lexicographic ordering is inappropriate.

At the same time, we contend that there are cases where balancing tradeoffs is either unnecessary or can be finessed. In some cases, the user may order domains and variables in accordance with a lexicographic order as we have defined it. In other cases, the user may be willing to impose such an order in order to obtain the representational and computational benefits. As a special case of this, the user may be able to specify a degree of ordering consistent with a small number of lexicographic orders, so that a small set of optimal solutions can be derived, with selection among them on other grounds. (This strategy has been discussed by Junker [9].) Finally, there are cases, such as the one given above, where tradeoffs can be finessed by imposing additional unary constraints. This is because such cases tend to occur when the values are extreme and, therefore, will have low preference values and may even have a negative valence.

Thus, while it is to be expected that there will be situations where simple lexicographic CSPs do not capture all aspects of the preference relation, if they are appropriate or if there is an acceptable total ordering on variables, then, as noted above, a representation can be provided that is transitive and complete, and has a sound axiomatic foundation.

4.2 Preference elicitation

When comparing alternative representations of preference, we must consider not only the soundness of the formal representation but also the degree to which the assumptions undergirding the representation can be met in practice. The latter may be called "soundness-in-use" as opposed to "soundness-in-conception". This issue is significant for the present work because observations of practice and careful empirical assessment both indicate that fewer inconsistencies arise when preference elicitation is based on a representation that is ordinal in character [39] [40] [41]. If inconsistencies occur, this means that the assumptions required by the model have not been met in practice. These results therefore show that even when cardinal representations (such as MAUT) are well grounded formally, it is harder to meet the conditions expressed by the axioms using this form of representation than is the case with ordinal representations, including lexicographic orderings.

Other empirical studies support the idea that simple rankings of differences and exclusion of alternatives on the basis of a few attributes are more natural activities than value and attribute scaling. Thus, there is a large body of work showing that most people use "noncompensatory" strategies, such as lexicographic ordering or elimination by aspects, when they must choose among alternatives that vary along several attributes [42]. This relative ease of use may explain why lexicographic orderings have been used in the past in decision-making applications, despite their inability to express tradeoffs directly, as noted by the main proponents of MAUT [6].

A useful aspect of the lexicographic CSP framework is that it allows one to represent ideal preference orderings in a clearcut fashion while still indicating why they cannot be realized. In contrast, these two aspects are not clearly distinguished in the usual soft constraint formulation. This is the principle of "decoupling" discussed in [24]. It is significant that, unlike the CP-net with feasibility constraints, which is a composite

system, the lexicographic CSP achieves a kind of "presentational" decoupling while retaining a representational coupling internally by combining preferences and feasibility constraints into a single (CSP) framework.

5 Solving Lexicographic CSPs with Ordinary CSP Algorithms

In addition to being well-grounded and perspicuous, the lexicographic CSP representation supports a variety of approaches to finding optimal solutions. As noted by Junker, with an ordinal representation such as this, preferences are not compiled into utility scales prior to search; instead, search methods are used that are based directly on the original preference orders [10]. This also obviates the need for global soft constraint representations such as those described in [31].

5.1 Basic strategies

The clear division between hard constraints and preferences in lexicographic CSPs allows us to use ordinary CSP search algorithms, which discard assignments that violate the former. This, of course, is not a practical strategy for weighted CSPs when there are no hard constraints, because in most cases if search were not bounded, no partial solutions could be discarded. This is also true for lex-VCSPs, and here one cannot usually order search in terms of constraint evaluations, since a single variable can be in the scope of more than one constraint. Thus, we can make the following statement:

If a lexicographic CSP has a solution α^* (cf. Definition 1), then this solution can be found and identified as optimal using a complete CSP algorithm with any order of instantiation of the variables.

In this case, a CSP algorithm is used to find all feasible solutions, and these can be compared lexicographically to find the optimal one. Essentially, this is a kind of filtered generate-and-test where we generate all feasible solutions in order to find the optimal one.

In addition, we have the following important special case:

If a lexicographic CSP is solved with a complete depth-first CSP algorithm using an ordering of variables and values consistent with the lexicographic ordering ("lexical ordering"), then α^* will be the first feasible solution found.

These facts give us considerable flexibility with regard to search strategies, so we can tailor search to fit the problem features. In particular, if there are many feasible solutions, then a simple lexical ordering of variables and values may be an excellent strategy. On the other hand, when it is not reasonable to use a lexical ordering, this will be because there are too few feasible solutions, and it is just this condition where good CSP heuristics, i.e. those designed to find feasible solutions quickly, may be useful. For these reasons, we begin our examination of solving methods with a study of ordinary CSP algorithms. In addition, since good heuristics should be transferrable to more sophisticated search techniques, results relevant for the latter can also be established.

5.2 Experimental tests

For experimental purposes, a total order can be simulated using random constraint satisfaction problems, where variables and values are represented as integers, 1 through n and 1 through $|d_i|$, respectively, where $|d_i|$ is the size of the ith domain. More specifically, the numerical labels of the variables and values serve as the required indices. In both cases, lower integer values represent preferred elements. Thus the solution, (1/1, 2/1, 3/1, ..., n/1), where x/y is the variable/value labelling, is the most preferred, (1/1, 2/1, 3/1, ..., n/2) the next-most preferred, etc. In keeping with the definition of a lexicographic ordering, a shift of value from k to $k + 1$ for a given variable represents a greater change in preference than a shift from k to $k + r$ for any variable with a higher index number.

In random problems of this type, the pattern of hard constraints has no relation to the preference ordering. These problems, therefore, have the usual pattern of difficulty, easy, hard, (relatively) easy, with respect to finding a *feasible* solution as either density or constraint tightness is varied while the other parameter is held constant [43]. Although we are interested in finding the best acceptable solution, since the number of feasible solutions changes dramatically as either density or tightness increases, we should still see a crossover at some point between lexically-ordered search and search based on ordinary CSP heuristics.

For our initial experiments, we used random problems with fixed values for number of variables, domain size, density and constraint tightness. The number of variables ranged from 10 to 40 in different experiments, and domain sizes were 10, 20, or 30. Although other densities were tested, in this section we restrict ourselves to density 0.5 for varying tightnesses.

In the initial experiments (Figures 2 and 3), algorithms for forward checking (FC) [44] and maintained arc consistency with AC-3 (MAC-3) [45] were written in C++ (by RH). In later experiments (Tables 1-5), MAC algorithms were coded in Lisp (by RJW). Extensive cross-comparisons ensured that results for search nodes and constraint checks were identical for versions of FC and MAC-3 written in the two languages. All experiments were run on a Dell Work Station PWS 330 running at 1.8 GHz. Since times, search nodes and constraint checks were always positively correlated for a given experiment, nodes expanded is used for most comparisons. Run time data is included in the Tables to give a more global measure of effort, since in this case the same measure could be used for both binary and nonbinary problems.

We compared performance of lexically-ordered search with that of two well-known variable and value ordering heuristics: (dynamic) minimum domain size variable ordering, where selection is based on current domain size, and (dynamic) minimum-conflicts value ordering, where values are selected that have the smallest sum of conflicts with values in adjacent future domains [46]. Although value ordering cannot make the all-solutions problem more efficient, it was of interest to determine whether such heuristics can find good solutions more quickly.

Figure 2 shows the search effort required by lexical ordering versus search with good CSP heuristics for forward checking. As expected, lexical search ordering is superior when there are many solutions. However, it was somewhat surprising that high efficiency is maintained over a considerable portion of the tightness range. In these

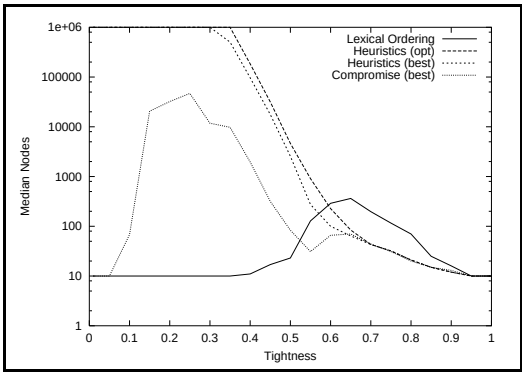


Fig. 2. Effort required, (i) to discover the best solution ("best"), (ii) to prove it optimal ("opt"). Forward checking algorithm, with either lexical ordering or good heuristics for ordinary CSPs. The effort required by a compromise ordering (min domain and lexical value) to discover the best solution is also shown. (The "opt" curve for this strategy is almost identical to that for the good-heuristics case.) Ten-variable problems, with fixed density = 0.5 and tightness varying in steps of 0.05. Sample size at each step is 50 problems. For tightness $\geq$ 0.65 problems have no solutions.

cases, ordinary heuristics are grossly inefficient either for finding the best solution or for determining optimality. However, despite the rigorous requirement that all solutions be tested, a crossover effect does occur as the number of solutions decreases, so that algorithms with good CSP heuristics can find the best solution and prove optimality with less effort than the lexical ordering. The crossover point occurs before the point where problems no longer have solutions, where lexical ordering must lose any advantage.

Figure 2 also shows the effort required by a compromise strategy (variable ordering by minimum domain size with ties broken by lexical ordering and lexically-ordered value selection) to find the best solution (without proving that it *is* the best). This strategy is much better at locating such solutions than the strategy based solely on good CSP heuristics, which makes the former suitable for either branch and bound or anytime procedures.

With MAC-3, the efficiency of lexical ordering is improved further, in relation to the other algorithms, so that for these small problems the crossover is no longer apparent. This apparent synergy between consistency maintenance and lexically ordered search may occur because arc consistency maintenance confers a greater benefit in the one-solution than the all-solutions case. However, the same crossover effect can be observed with larger problems, and becomes more pronounced as domain size increases (Figure 3, Table 1). Note that the y-axis in Figure 3 is logarithmic, so that after two corresponding curves cross, there is an interval of tightness where the heuristic method improves on the lexical ordering by a factor of 2-4.

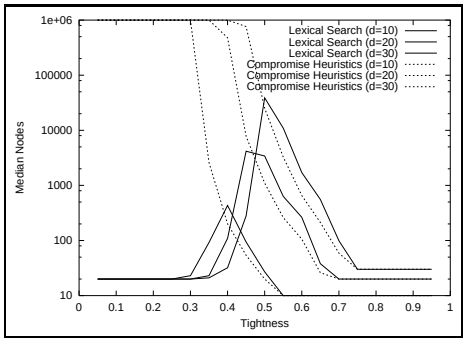


Fig. 3. Effort required by MAC-3 to obtain provably optimal solutions with lexical and compromise ordering. Twenty-variable problems with density = 0.5 and domain size = 10, 20 and 30. Tightness varied in steps of 0.05 for each curve. Sample size at each step is 50.

6 More Advanced Solving Methods

6.1 Branch and bound with CSP heuristics

An alternative form of search is to combine branch and bound with CSP heuristics involving dynamic variable ordering. (Obviously, lexically ordered search cannot be improved in this manner, since it orders search in terms of an implicit cost function.) We have already described a cost function for the weighted CSP version of lexicographic CSPs that uses base arithmetic, but for lexicographic CSPs we do not need to calculate the actual values of this function, which are quite large for any but small problems. Instead, we simply compare successive values following the lexical variable ordering until we encounter a difference.

Specifically, suppose that variable X_i is the variable currently being considered for instantiation, and this variable is the kth most important variable in the ordering. To evaluate the current partial solution, we start from the first variable in the lexical ordering. If a variable has an assignment, we check this against its instantiation in the best assignment found so far; if it does not yet have an assignment, we check the best remaining value in its domain against the best assignment. In either case, if we encounter a value greater than the best found so far, then search can back up. Pseudocode for this algorithm is shown in Figure 4.

The efficiency of this procedure could suffer when combined with variable ordering heuristics, if variables of high priority occur late in the search order, and good values are still available. Despite this potential limitation, this form of branch and bound does well in comparisons with other methods (Table 1). Not only does it give further improvements in performance when there are few solutions, but its performance degrades more slowly than the ordinary CSP algorithm when the number of feasible solutions increases, and in these cases it avoids checking a large proportion of solutions. It is also much less sensitive to increases in domain size than lexical ordering. It, therefore, can be used with less concern about the exact properties of the problem.

```
initialise values in best-solution to BIGVAL

lexico-bnb (partial-solution, remaining-variables)
        if remaining-variables ≡ nil
                save partial-solution as new best-solution
                and continue          //backtrack
        else
                select next-variable and remove from remaining-variables
                for each value in its ordered domain
                   if instantiating next-variable with this value leads to an arc consistent problem
                      and
                      bounds-check(next-variable, next-value) returns true //under bound
                            lexico-bnb (new-partial-solution, remaining-variables)
                continue              //backtrack

bounds-check (candidate-var, candidate-value)
        underbound = true; comparison-succeeded = false
        while variables remain to be compared & not comparison-succeeded
                select next-variable = most important unchecked variable
                get value best-value for this variable from current best-solution
                if next-variable == candidate-var
                   curr-assign = candidate-value
                else if next-variable is instantiated
                   curr-assign = current assignment of next-variable
                else                  //smallest value is most-preferred value
                   curr-assign = smallest value in current domain of next-variable

                                      //perform comparisons
                if curr-assign > best-value
                   comparison-succeeded = true
                   underbound = false
                else if curr-assign < best-value
                   comparison-succeeded = true

        if comparison-succeeded
                if underbound
                        return true
                else
                        return false
        else
                return true
```

Fig. 4. Branch and bound pseudocode for lexicographic CSPs.

6.2 A specialised lexicographic CSP algorithm

Another strategy for search algorithms in this domain is to devise procedures that are sensitive to the properties of a lexicographic order. In this vein, we have implemented a specialized iterative algorithm for lexicographic CSPs that we call "staged lexical search". Search is done repeatedly, in each case until the first solution is found, and for each repetition or stage of search, one more variable is chosen in lexical order, i.e. in accordance with the importance ordering. Values for this variable are chosen according to the preference ordering for this domain, starting with the most-preferred value.

Thus, in Stage 1 we first select variable X_1 according to the lexical (importance) ordering, and then use any heuristic to select the others. When we have found a feasible solution, we know that the assignment for X_1 is optimal, so we retain it for the remainder of search. In Stage 2, we first select variable X_2, so the first feasible solution found will include an optimal assignment for this variable. And so forth. Pseudocode for the

staged lexical algorithm is shown in Figure 5. Although developed independently, this algorithm is a special case of preference-based search [14], where the criteria on which search is based form a total order.

As noted earlier, methods for lexicographic ordering in multiobjective optimisation generally use a weighting scheme, in contrast to adding successive constraints according to the importance ordering of the criteria. However, a specialised technique used to find Pareto-optimal solutions, called the epsilon-constraints method, is based on a similar strategy to that used for preference-based search. The epsilon-constraint problem is defined as follows:

$$\min f_j(x)$$

subject to

$$f_k(x) \ \leq \epsilon_k, k = 1, \ldots, n \ \ k \neq j$$

For CSPs with ordered domains, this corresponds to finding a minimal value in domain j, while allowing a choice of values in domains $j+1, \ldots, n$, subject only to some minimal difference from the optimal. In the present case, ϵ_k $(k > j)$ can be considered to be set high enough to allow any domain value to be chosen. However, since the purpose of the epsilon-constraint method is to find solutions that belong to the Pareto frontier, it does not involve successive addition of constraints in the manner of the CSP methods. (A basic description of this method is in [47]; see [48] for a recent discussion.)

```
k = 1
while k ≤ n
    level = k
    while level ≤ n                 //search for next solution
        if level == k
            select kth variable in importance order
        else
            select next-variable according to some heuristic
        while values remain and viable assignment not found
            if level == k
                select value according to preference ordering and propagate
            else
                select value according to some heuristic and propagate
        if all assignments have failed at this level
                //can't happen if level == k and there is a solution
            backtrack and set level = level - 1
        else
            level = level + 1

    save assignment made at level k                 //this is optimal
    k = k + 1
```

Fig. 5. Pseudocode for staged lexical algorithm for lexicographic CSPs.

In the form just described, this algorithm also compares well with other methods (Table 1). Not only does it do almost as well as branch and bound when there are few solutions, but it never does much worse than simple lexical ordering for problems with many solutions. In addition, mean runtimes for staged lexical were almost identical to those for branch and bound in cases where there were few or no solutions (when there were many solutions, staged lexical was faster). It may, therefore, be the algorithm of choice if only one method is desired for all species of problems. It may also be possible to improve the algorithm somewhat since often after a certain stage, the same solution is found repeatedly.

6.3 Further experimental comparisons

In this section, we present results based on other problem sets. The first set of results is for random problems with a larger number of variables. The second is for problems with heterogeneous structure, consisting of an easy subproblem with loose constraints and a hard subproblem with tighter constraints. To facilitate comparison with previous tests, these problems have similar size and density to the random 20-variable problems. The third set of results is for problems derived from a real-world configuration problem obtained from the Configuration Benchmarks Library maintained by the Computational Logic and Algorithms Group at the University of Copenhagen (ESVS Benchmark #7 [3]).

For 40-variable problems with the same domain size and density as before, the same trends appear, but the differences are magnified. In particular, for problems in the critical complexity region, the branch and bound and staged lexical algorithms are an order of magnitude better on average than the basic lexical ordering. Although the compromise method is also better than simple lexical ordering for the hard problems (mean nodes = 280,514, using the min domain heuristic), since it is clearly dominated by the advanced methods, results for this method are omitted from this and later tables. Interestingly, the difference between the two advanced methods is now quite pronounced as soon as one enters the easy portion of the complexity space. (For purposes of comparison, we include data for these problems based on ordinary CSP search for one solution using the same variable ordering heuristic. This serves to show how effective the advanced methods are for problems that are also hard as CSPs.)

Problems with heterogeneous structure were like the original set of homogeneous random problems in having 20 variables, with domains of size 10, and the overall graph density was 0.50. There were two components, each with ten variables: an ‘easy’ component with very loose constraints (tightness = 0.05) and a ‘hard’ component with tighter constraints (either 0.60 or 0.65). These components were linked by constraints also with a tightness of 0.05. Within each component as well as for the links, the proportion of edges was 0.5. These problems were of a type that has been called “composed” problems [49].

For the experiments described here, the importance ordering was such that the ten variables in the easy component were designated as the ten most important. (If the ordering favored variables in the hard component, the results were similar to those given for

[3] http://www.itu.dk/research/cla/externals/clib/esvs.pm

Table 1. Search Efficiency Comparisons

	hard problems			easy problems		
domain size	10	20	30	10	20	30
tightness	0.35	0.45	0.50	0.30	0.40	0.45
lexical						
median nodes	149	4631	30,959	26	85	292
mean nodes	347	9084	121,786	37	174	1027
mean solns	1	1	1	1	1	1
mean runtime	0.53	20.0	415.9	0.04	0.28	2.2
CSP compromise						
median nodes	1402	7404	26,053	373,260	-	-
mean nodes	1599	7927	29,858	410,654	-	-
mean solns	193	35	8	196,731	-	-
mean runtime	1.4	19.6	100.6	30.3	-	-
branch and bound						
median nodes	165	1238	6252	322	945	2381
mean nodes	217	2020	9395	380	1189	2991
mean solns	2	1	1	6	6	6
mean runtime	0.41	5.2	32.5	0.41	2.2	6.9
staged lexical						
median nodes	325	1511	7152	230	338	506
mean nodes	390	2330	9778	237	375	607
mean solns	20	20	20	20	20	20
mean runtime	0.39	5.4	31.7	0.14	0.43	0.98

Notes. Twenty-variable problems, sample size 100. MAC algorithm. “hard problems” are near the critical complexity peak for lexical ordering. “easy problems” are near the edge of the hard region for lexical (cf. Figure 3). “solns” is number of feasible solutions found during the entire search; for CSP compromise this is the total number of solutions per problem. Branch and bound and staged lexical algorithms employed the compromise ordering. Here and elsewhere run times are in seconds.

random problems.) In this case, search using a basic lexical ordering does very poorly because of excessive thrashing, since conflicts are not uncovered until search is deep in the tree. For this algorithm, search could not be run to completion for all problems, so it was only possible to derive a lower bound (based on total accumulation of nodes across a problem-set). At the same time, the advanced methods when given an appropriate heuristic were able to solve these problems efficiently. (The heuristic used in this case was the FF2 heuristic of [50], a Brelaz-like heuristic which is sensitive to relative constrainedness of different parts of the problem.) Here, the staged lexical algorithm outperformed branch and bound significantly and did not encounter any difficult problems. As a result, the former sometimes outperformed the standard lexically ordered search by at least five orders of magnitude with respect to the mean.

The success of these algorithms was not simply due to the use of an appropriate heuristic, since when FF2 was used with the all-solutions “compromise” strategy de-

Table 2. Search Comparisons for Larger Problems

	hard problems	easy problems
tightness	0.20	0.15
CSP-1 sol		
median nodes	38,054	42
mean nodes	62,005	55
mean solns	1	1
mean runtime	323.9	0.10
lexical		
median nodes	375,040	68
mean nodes	817,710	120
mean solns	1	1
mean runtime	5768.6	0.24
branch and bound		
median nodes	51,301	160,480
mean nodes	76,513	4,385,167
mean solns	1.4	41
mean runtime	444.3	3294.1
staged lexical		
median nodes	64,526	969
mean nodes	83,612	1007
mean solns	40	40
mean runtime	526.9	2.0

Notes. Forty-variable problems, domain size 10, density 0.5 sample size 100. For reference, "CSP-1 sol" shows results for ordinary CSP search, using min domain and lexical value ordering. Other notes as in Table 1.

scribed earlier, the algorithm could not be run to completion on any of these problems using a 10 million node cutoff. This is not surprising, since these problems typically have millions of feasible solutions.

The original ESVS configuration problem has 26 variables (with domain sizes ranging from two to 61) and 11 constraints with a maximum arity of five. In constructing preference problems, we first discarded six variables that were disconnected. (They were not of interest in the present context since the most preferred value is always available). Then the remaining 20 variables were labelled lexically (following the order of listing in the source file), and 100 Lexicographic CSPs were constructed with different lexical orders on these variables by renaming the variables according to random permutations of the original labels. (This particular strategy allowed us to continue using the lexical labels as indicators of importance in the solver code.) Doing this, we were able to test our algorithms over a large sample of possible importance orderings with respect to configuration components. For each algorithm, the solver code was the original code updated with code for generalised arc consistency; hence the algorithm employed a generalised form of MAC3.

Table 3. Search Comparisons for Heterogeneous Problems

	tightness of hard component	
	0.60	0.55
lexical		
median nodes	138,999	27
mean nodes	> 10M	> 7M
mean solns	1	1
branch and bound		
median nodes	389	1268
mean nodes	109,332	411,791
mean solns	3.0	5.3
mean runtime	7.3	23.3
staged lexical		
median nodes	254	243
mean nodes	259	247
mean solns	20	20
mean runtime	0.18	0.13

Notes. Twenty-variable problems, domain size 10, density 0.5, with two components of size 10: an 'easy' part with constraints of tightness = 0.05 and a 'hard' part with tightness as indicated in the column headers. The two components are linked by constraints with tightness = 0.05. The density value given above also holds within each component and for the proportion of possible links between them. "M" means million. Sample size 100. Other notes as in Table 1.

We generated three sets of problems using the approach just described. The first set of (easy) problems was generated from the original ESVS problem. For the second set, the original problem was altered by adding new constraints so that the constraint graph was a single connected component, and by changing some of the relations to reduce difference in support for domain values, especially in binary domains, and finally by discarding tuples associated with solutions until the problem became difficult for simple lexical search. The 100 most difficult problems were then culled from a set of 1000 generated according to the methods described in the previous paragraph. Finally, a third base problem was derived by duplicating the variables and constraints from the second (and relabelling the variables) and adding a constraint that connected the two sub-problems via corresponding variables with binary domains, that included only tuples with distinct values (i.e. (0, 1) and (1, 0)). Then 100 problems were generated according to the methods described in the previous paragraph.

The results of these experiments are shown in Table 4. The basic pattern of results corresponds to those of previous experiments. Problems based on the original ESVS problem have many feasible solutions (3.1 million in this case), and here branch and bound algorithms are relatively inefficient. When feasible solutions are harder to find, then the advanced algorithms outperform simple lexical search by up to an order of magnitude. In addition, the disparity between means and medians indicates that while most problems are still easy, certain importance orderings wreck havoc with the simple lexically-ordered search, leading to a high mean search effort.

Table 4. Search Comparisons for Configuration Problems

	original (easy) ESVS	altered ESVS	large altered ESVS
lexical			
median nodes	20	294	41
mean nodes	20	899	98,707
mean solns	1	1	1
mean runtime	0.01	0.25	32.3
branch and bound			
median nodes	789	305	3684
mean nodes	41,294	418	21,170
mean solns	3.1	2.7	6.0
mean runtime	23.9	0.13	8.9
staged lexical			
median nodes	210	363	851
mean nodes	210	536	6062
mean solns	20	20	40
mean runtime	0.10	0.17	2.2

Notes. Each set of problems consists of 100 different importance orderings derived from an original configuration problem. Branch and bound and staged lexical search employed min domain variable ordering.

6.4 Problems with embedded lexicographic orders

In a final set of experiments, we considered problems for which preferences are restricted to a subset of the variables in the CSP representation, that we call "embedded preference orderings". Cases like these may arise often in practice, where in addition to variables representing attributes whose values are subject to user selection, there are variables representing constraints of various kinds that the user has no preferences about, such as physical constraints on design, features not visible to the user, etc. These experiments employed the 20-variable random and heterogeneous problems used in previous tests, but now only the first 5 or 10 variables in the lexical ordering were used as the basis for preferences. With the heterogeneous problems, the embedded ordering was in the easy component; this represents the case where preferences are associated with less constrained parts of a problem that contains other, difficult subproblems. (Note that the alternative case is already modelled by the embedded orderings in the homogeneous random problems.) For algorithms that involved lexical ordering, the procedure switched to an appropriate heuristic once the variables in the preference ordering were instantiated: minimum domain size for the random problems and FF2 for the composed problems.

A selection of the results is shown in Table 5. Differences among algorithms are for the most part similar to those observed in previous experiments, although the branch and bound is more efficient than before on the heterogeneous problems when only 25% of the variables are in the embedded lexicographic order. Most significantly, the spe-

cialised algorithms continue to outperform the standard lexical ordering on these problems even when only a small proportion of the variables are in the embedding.

Table 5. Search Comparisons for Problems with Embedded Lexicographic Orders

problems	random <20,10,.50>		composed 10-10	
tightness	0.35	0.30	0.60	0.60
embedd. size	10	10	10	5
lexical				
median nodes	149	26	122,480	29
mean nodes	347	37	> 10M	12,000
mean solns	1	1		1
mean runtime	0.54	0.03	-	14.3
branch and bound				
median nodes	171	373	535	209
mean nodes	220	422	109,480	213
mean solns	4	27	6	1
mean runtime	0.27	0.30	9.3	0.21
staged lexical				
median nodes	280	185	209	132
mean nodes	345	192	214	135
mean solns	10	10	10	5
mean runtime	0.35	0.10	0.23	0.16

Notes. Twenty-variable problems used in previous experiments (cf. Tables 1 and 3). For composed problems, tightness value is for the hard component. (The embedded ordering is in the easy component.) "embedd. size" is the number of variables in the preference ordering. "M" means million. Sample size 100. Other notes as in Table 1.

7 Conclusions

By combining lexicographic orderings with the classical CSP formalism, we can represent both preferences and hard feasibility constraints together in a manner that is formally well-grounded, easily comprehended, and amenable to robust techniques for problem solving. A constraint-based representation may also make it possible to extend the application of lexicographic orderings in a significant fashion, by precluding some types of tradeoffs through feasibility constraints.

The completeness of lexicographic orderings allows more comparisons between alternatives than is possible with other qualitative preference formulations, such as CP-nets. At the same time, such comparisons are often much more straightforward. Desirable features like conditional preferences can also be incorporated into this form of representation.

Problem solving methods based on the ordinal properties of lexicographic CSPs are often very efficient. In our studies of search algorithms for lexicographic CSPs, we de-

vised several search strategies that are effective for the task of finding optimal solutions, including a branch and bound strategy and a specialized algorithm that takes advantage of the lexicographic ordering to find successive optimal assignments in a series of iterations. These procedures sometimes outperformed a standard form of search that directly follows the lexicographic ordering by several orders of magnitude.

References

1. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint solving and optimization. J. ACM **44** (1997) 201–236
2. Bistarelli, S., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G., Fargier, H.: Semiring-based csps and valued csps: Frameworks, properties and comparison. Constraints **4** (1999) 199–240
3. Schiex, T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: Hard and easy problems. In: Proc. Fourteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1995) 631–637
4. Majumdar, T.: The Measurement of Utility. Macmillan (1966)
5. Doyle, J., Thomason, R.H.: Background to qualitative decision theory. Artificial Intelligence Magazine **20** (1999) 55–68
6. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives. Preferences and Value Tradeoffs. Cambridge (1993)
7. Saaty, T.L.: The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making. In Figueira, J., Greco, S., Ehrgott, M., eds.: Multiple Criteria Decision Analysis. Springer (2005) 345–407
8. Brewka, G.: Preferred subtheories: An extended logical framework for default reasoning. In: Proc. Eleventh International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1989) 1043–1048
9. Junker, U.: Preference-based search and multi-criteria optimization. Ann. Operat. Res. **130** (2004) 75–115
10. Junker, U.: Preference-based problem solving for constraint programming. In Fages, F., Rossi, F., Soliman, S., eds.: Recent Advances in Constraints: 12th Annual ERCIM International Workshop on Constraint Solving and Constraint Logic Programming - CSCLP 2007. Revised Selected Papers. LNAI 5129. (2008) 109–126
11. Boutilier, C., Brafman, R.I., Hoos, H.H., Poole, D.: Reasoning with conditional ceteris paribus preference statements. In: Proc. Fifteenth Annual Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann (1999) 71–80
12. Wallace, R.J., Wilson, N.: Conditional lexicographic orders in constraint satisfaction problems. In Beck, J.C., Smith, B.M., eds.: Proc. Third International Conference on Integration of AI and OR Techniques in Constraint Programming - CPAIOR 2006. LNCS. No. 3990, Berlin, Springer (2006) 258–272
13. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H., Poole, D.: CP-nets: A tool for representing and reasoning with *ceteris paribus* preference statements. Journal of Artificial Intelligence Research **21** (2004) 135–191
14. Junker, U.: Preference-based search and multi-criteria optimization. In: Proc. Eighteenth National Conference on Artificial Intelligence, AAAI Press (2002) 34–40
15. Fishburn, P.C.: Lexicographic orders, utilities and decision rules: A survey. Management Sci. **20** (1974) 1442–1471
16. Fargier, H., Lang, J., Schiex, T.: Selecting preferred solutions in fuzzy constraint satisfaction problems. In: Proc. First European Conference on Fuzzy and Intelligent Technologies - EUFIT'93. (1993) 1128–1134

17. Borning, A., Freeman-Benson, B., Wilson, M.: Constraint hierarchies. Lisp Symbol. Computat. **5** (1992) 223–270
18. Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Walsh, T.: Global constraints for lexicographic orderings. In Hentenryck, P.V., ed.: Principles and Practice of Constraint Programming - CP2002, LNCS 2470, Springer (2002) 93–108
19. Freuder, E.C., Wallace, R.J., Heffernan, R.: Ordinal constraint satisfaction. In: Fifth International Workshop on Soft Constraints - SOFT'02. (2003)
20. Brafman, R.I., Domshlak, C.: Introducing variable importance tradeoffs into CP-nets. In: Proc. Eighteenth Annual Conference on Uncertainty in Artificial Intelligence. (2002) 69–76
21. Brafman, R.I., Domshlak, C., Shimony, E.: Graphical modeling of preference and importance. Journal of Artificial Intelligence Research **25** (2006) 389–424
22. Domshlak, C., Brafman, R.I.: CP-nets - reasoning and consistency testing. In: Proc. Eighth Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann (2002) 121–132
23. Goldsmith, J., Lang, J., Truszczynski, M., Wilson, N.: The computational complexity of dominance and consistency in CP-nets. In: Proc. Nineteenth International Joint Conference on Artificial Intelligence. (2005) 144–149
24. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H., Poole, D.: Preference-based constrained optimization with CP-nets. Computational Intelligence, Special Issue on Preferences **20** (2004) 137–157
25. Wilson, N.: Consistency and constrained optimisation for conditional preferences. In: Proc. Sixteenth European Conference on Artificial Intelligence. (2004) 888–892
26. Wilson, N.: An efficient upper approximation for conditional preference. In: Proc. Seventeenth European Conference on Artificial Intelligence. (2006) 472–476
27. Wilson, N.: Extending CP-nets with stronger conditional preference statements. In: Proc. Nineteenth National Conference on Artificial Intelligence. (2004)
28. Wallace, R.J., Wilson, N.: Conditional lexicographic orders in constraint satisfaction problems. Annals of Operations Research special issue entitled "Constraint Programming, Artificial Intelligence and Operations Research" (2009)
29. Ehrgott, M., Wiecek, M.M.: Multiobjective programming. In Figueira, J., Greco, S., Ehrgott, M., eds.: Multiple Criteria Decision Analysis. State of the Art Surveys. Springer (2005) 667–722
30. Gavanelli, M.: An algorithm for multi-criteria optimization in csps. In van Harmelen, F., ed.: Fifteenth European Conference on Artificial Intelligence-ECAI 2002, IOS Press (2002) 136–140
31. Joseph, R.R., Chan, P., Hiroux, M., Weil, G.: Decision-support with preference constraints. Eur. J. Oper. Res. **177** (2007) 1469–1494
32. Ehrgott, M.: Multicriteria Optimization. 2nd edn. Springer (2005)
33. Korhonen, M.: Interactive methods. In Figueira, J., Greco, S., Ehrgott, M., eds.: Multiple Criteria Decision Analysis. State of the Art Surveys. Springer (2005) 641–665
34. Luce, R.D., Suppes, P.: Preference, utility and subjective probability. In Luce, R.D., Bush, R.R., Galanter, E., eds.: Handbook of Mathematical Psychology. Volume 3. Wiley (1965) 249–410
35. Fishburn, P.C.: Utility theory for Decision Making. Wiley (1970)
36. Fishburn, P.C.: Axioms for lexicographic preferences. Rev. Econ. Stud. **42** (1975) 415–419
37. Plott, C.R., Little, J.T., Parks, R.P.: Individual choices when objects have "ordinal" properties. Rev. Econ. Stud. **42** (1975) 403–413
38. Larichev, O.I.: Psychological validation of decision methods. J. Appl. Syst. Anal. **11** (1984) 37–46
39. Arrow, K.J., Raynaud, H.: Social Choice and Multicriterion Decision-Making. MIT (1986)

40. Borcherding, K., Eppel, T., von Winterfeldt, D.: Comparison of weighting judgments in multiattribute utility measurement. Managment Sci. **37** (1991) 1603–1619
41. Olson, D.L., Moshkovich, H.M., Schellenberger, R., Mechitov, A.I.: Consistency and accuracy in decision aids: Experiments with four multiattribute systems. Decis. Sciences **26** (1995) 723–746
42. Payne, J.W., Bettman, J.R., Johnson, E.J.: The Adaptive Decision Maker. Cambridge (1993)
43. Cheeseman, P., Kanefsky, B., Taylor, W.M.: Where the *really* hard problems are. In: Proc. Twelth International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1991) 331–337
44. Haralick, R.M., Elliott, G.L.: Increasing tree search efficiency for constraint satisfaction problems. Artificial Intelligence **14** (1980) 263–313
45. Sabin, D., Freuder, E.: Contradicting conventional wisdom in constraint satisfaction. In: Proc. Eleventh European Conference on Artificial Intelligence, Wiley (1994) 125–129
46. Frost, D., Dechter, R.: Look-ahead value ordering for constraint satisfaction problems. In: Proc. Fourteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1995) 572–578
47. Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making. Theory and Methodology. Dover (1983)
48. Laumanns, M., Thiele, L., Zitzler, E.: An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. Eur. J. Operat. Res. **169** (2006) 932–942
49. Lecoutre, C., Boussemart, F., Hemery, F.: Backjump-based techniques versus conflict-directed heuristics. In: Proc. Sixteenth International Conference on Tools with Artificial Intelligence-ICTAI'04. (2004) 549–557
50. Smith, B.M., Grant, S.A.: Trying harder to fail first. In: Proc. Thirteenth European Conference on Artificial Intelligence—ECAI'98, John Wiley & Sons (1998) 249–253