Ordinal Constraint Satisfaction

Eugene C. Freuder, Richard J. Wallace and Robert Heffernan *

Cork Constraint Computation Centre
University College Cork, Department of Computer Science, Cork, Ireland
e.freuder-,r.wallace-,r.heffernan@4c.ucc.ie

Abstract. We propose a new CSP formalism that incorporates hard constraints and preferences so that the two are easily distinguished both conceptually and for purposes of problem solving. Preferences are represented as a lexicographic order over variables and domain values, respectively. Constraints are treated in the usual manner. Therefore, these problems can be solved with ordinary CSP algorithms, with the proviso that complete algorithms cannot terminate search after finding a feasible solution, except in the important case of heuristics that follow the preference order (lexical order). We discuss the relation of this problem representation to other formalisms that have been applied to preferences, including soft constraint formalisms and CP-nets. We show how algorithm selection can be guided by work on phase transitions, which serve as a useful marker for a reversal in relative efficiency of lexical ordering and ordinary CSP heuristics due to reduction in number of feasible solutions. We also consider branch and bound algorithms and their anytime properties. Finally, we consider partitioning strategies that take advantage of the implicit ordering of assignments in these problems to reduce the search space.

1 Introduction.

An important contemporary 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.

In trying to characterize such problems, a common approach in applied decision-making is to construct hierarchies 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. After that, preference functions are constructed that take into account priorities and tradeoffs among these attributes [KR93] [vWE86].

In this paper we describe an approach that represents such decision problems as a kind of constraint satisfaction problem (CSP). We model preferences in terms of orderings among solutions to a CSP. If we keep to this assumption, a simple and natural way to represent preferences in CSPs is as a *lexicographic* ordering of assignments both in terms of the variables and values assigned. Here, we consider variable selection as the primary factor and value selection as secondary. This means that a good assignment for

* This work received support from Science Foundation Ireland under Grant 00/PI.1/C075.

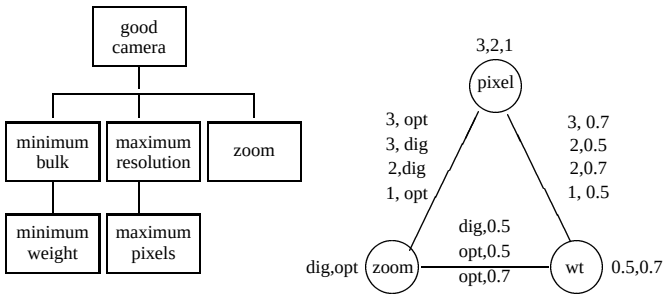


Fig. 1. Multiattribute decision problem, represented by a hierarchy of objectives on the left with measurable attributes at the lowest level and by an ordinal CSP on the right, which also incorporates tradeoffs and restrictions as hard constraints. (Constraints are shown as viable tuples.)

a more-preferred variable is more important than a good assignment for a less-preferred variable in deciding the overall ranking of solutions. We call such problems *ordinal* constraint satisfaction problems.

This type of metric may be perspicuous for many applications. For example, a customer may have 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 number preferred in the first case and a smaller in the second. In addition, let us suppose that a digital zoom is preferred over an optical zoom, and this attribute has a priority between pixels and weight.

Putting this into the form of an objectives hierarchy gives the result shown on the left of Figure 1. To round out this example we will suppose, further, that a larger number of pixels is associated with greater weight, and that there are also restrictions, or tradeoffs, between pixels and zoom-type and zoom-type and weight. All this can be represented by the ordinal CSP on the right of Figure 1.

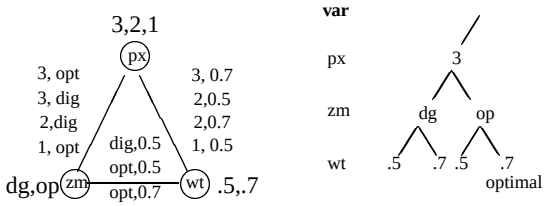
A major benefit of this approach is that preferences and hard constraints are cleanly separated. One result is that, when constraint satisfaction problems have a mixture of hard (must be) and soft (prefer) constraints, we can choose to order search so as to maximize our chances of finding a solution to the hard constraints quickly or to maximize the chances of finding an optimal solution for the soft constraints quickly. Or we can choose a compromise strategy. The best search order strategy may depend on the mix of hard and soft constraints.

Ordinal CSPs also have the nice property that if we use the preference orderings as our variable and value orderings, the first solution to the hard constraints that we find will be the optimal one. Unfortunately, these kinds of orderings are likely to be give poor performance in many cases because they disregard the properties that normally make search efficient, properties such as smallest domain size, maximum degree, etc. Fortunately, these problems also have the property that if we consider any domain in isolation, or even a subset of that domain, the preference ordering will not change when

other variables are given an assignment, a property known as “preferential independence” [KR93]. This means that we can use dynamic CSP heuristics for variable and value ordering, and we can also ‘mix’ these heuristics with selection based on preferences among values in single domains without having to alter our selections because of dependencies between preferences.

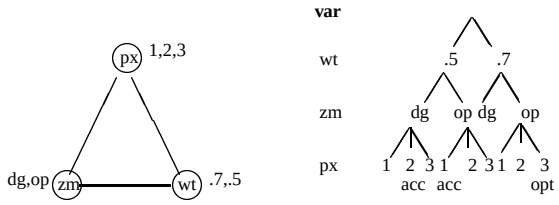
These ideas can be illustrated with the example given in Figure 1. In this example, the priority ordering of attributes is pixels > zoom > weight, and the values for pixels in order of preference are 3, 2, and 1 (megapixels), while for zoom they are digital and optical, and for weight 0.5 and 0.7. This means the most-preferred solution is 3 for pixels, digital for zoom and 0.5 for weight. Unfortunately (3, digital, 0.5) does not satisfy all the hard constraints. In fact, the best acceptable solution is (3, optical, 0.7).

By using the lexicographic ordering to make search decisions, we can proceed through successively less preferred solutions until we find one that satisfies the hard constraints. The inset below shows the backtrack search tree that results from this ordering (For reference, the constraint graph and value orderings [from left to right] are shown on the left. As usual, the depth-first search tree is expanded downward and from left to right.):



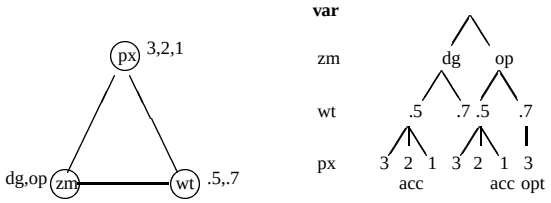
Here, we know that the first acceptable solution (i.e. one that satisfies all hard constraints) is optimal, so search can stop at this point. (Seven nodes needed to be expanded in this search.)

However, this ordering is not necessarily well chosen for finding solutions to the hard constraints quickly. If we use a “minimum domain” variable ordering (with ties broken arbitrarily), and a “minimum conflicts” value ordering – and assuming the values in each domain are in some unrelated order – we get the following search tree leading up to the first solution (Again, value orderings are shown with the graph on the left.):



This finds an acceptable solution sooner (after four nodes are expanded). However, this search order tends to bias search against finding the optimal solution quickly; the latter occurs after expansion of 15 nodes. (At this point all possibilities have been checked, so optimality has also been proven.)

Now suppose we use a compromise search order. We use minimum domain variable ordering, but break ties using the preference order, and we use the preference order for value ordering:



This also finds acceptable solutions quickly; more importantly, it finds the optimal solution a bit sooner (13 nodes expanded), although not as quickly as the lexical ordering. But it still requires as much search to prove optimality as the good heuristics did. Thus, there are indications even in this toy example that, while lexical ordering should predominate for easy problems, good heuristics may be more efficient when there are few feasible solutions to begin with, and the compromise ordering may be an effective anytime strategy. These indications are borne out, as shown later in the paper.

Although lexicographic orderings have a venerable history in the study of preference in such areas as economics and decision making [Fis68], there has been very little work on applying this idea to constraint satisfaction problems, and little work on algorithms for solving these problems. In the context of symmetry breaking, [FHK$^+$02] have studied consistency algorithms for constraints that require lexicographic ordering with respect to sets of variable-vectors. In addition, the atom-bound probability formalism used to model credibility or belief is also based on the notion of "overwhelming advantage" of higher-priority elements over any set of elements of lower priority [Sno94] [BDP97]. As our introductory example indicates, combining this idea with constraint based reasoning gives a useful representation for many decision problems that is also amenable to a variety of algorithms and heuristics that have been developed in connection with ordinary CSPs.

The next section discusses relations between ordinal CSPs and other formalisms, specifically, soft constraints and CP-nets. Section 3 considers the complexity of ordinal CSPs. In Section 4 we compare different CSP algorithms and heuristics applied to ordinal CSPs with varying numbers of acceptable solutions. Section 5 describes and evaluates a branch and bound algorithm for ordinal CSPs. Anytime properties of such algorithms are studied in Section 6. In Section 7 we consider preprocessing strategies for these problems that partition the problem space to zero in on good solutions. Section 8 gives conclusions.

2 Relations to Other Formulations.

2.1 Formal definition.

An ordinal CSP can be defined as a tuple, $P = (X, D, C, >)$, where
X = a set of n variables,
D = a set of domains, $D_1, D_2, ..., D_n$ where each D_i is associated with a member of

X, X_i, and consists of a set of k_i values, any one of which can be assigned to X_i,
C = a set of constraints, where each constraint C_j is associated with a subset of variables in X and constitutes a relation based on Cartesian product of the domains of the variables in this subset,
$>$ = a total order defined on n-dimensional vectors of variable assignments, $A = (a_1, a_2, ..., a_n)$, where $a_1 \in D_1$,$a_2 \in D_2$, ... and where assignment $A_i > A_j$ iff $a_{ki} > a_{kj}$ for the kth variable in the vector, while $a_{li} = a_{lj}$ for all $l < k$ in the vector.

A solution to an ordinal CSP is an assignment A^* such that

(i) A^* is a satisfying assignment, that is, it is consistent with, or satisfies, all constraints in C

(ii) for any satisfying assignment A, $A^* \geq A$.

2.2 Ordinal CSPs and soft constraints.

In constraint-based reasoning, preferences are typically modeled as soft constraints, in which failure to satisfy a constraint serves to deprecate the offending values, but does not lead to outright exclusion. At present, the leading formalisms are the valued and semiring CSPs, in which evaluations are attached to domain values as well as to either constraints or constraint tuples [BMR97] [SFV95]. 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 maximum sum or the largest minimum value associated with any value or tuple in an assignment.

In many situations, ordinal CSPs may be a useful alternative to soft constraint representations. In the first place, it is natural to rank the different values of an attribute and then to rank the attributes themselves in some fashion. Otherwise, it is difficult to see how organizing objectives into hierarchical structures could have proven as useful as it has [KR93] [vWE86]. There is also direct support in the psychology of choice, which has produced evidence that people process complex decisions by successively eliminating attributes (called "aspects") according to some priority ordering [Tve72]. In this case, it may even be possible to establish a lexicographic preference ordering during the course of problem solving, by allowing the user to follow his or her natural strategy of eliminating successive attributes.

We also note that indifference between alternatives can be handled in a straightforward manner in this framework. If there are indifference sets within the preference ordering, we can use an arbitrary order for purposes of problem solving; then, if one of k indifferent values gives a solution while the remaining values do not, we can rearrange the order. This is important, because it means that we do not require that users must choose decisively among all alternatives. Here, we make the assumption that the user can distinguish between all variables and values, so indifference is a lack of preference rather than a difficulty in distinguishing alternatives. In this case, we do not need to concern ourselves with problems due to intransitivity of the indifference relation that arise almost of necessity in the latter case [Luc56]. (In fact, it is difficult to see how prob-

lems of discrimination can arise in a prescriptive context where the user is specifying preferences to begin with.)

A major strength of the ordinal CSP approach 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 of a problem are blurred in the usual soft constraint formulation. As a result, while it is to be expected that there will be situations where ordinal CSPs are insufficient, in other cases this may well be a more appealing approach by virtue of its greater clarity and simplicity.

In addition, ordinal CSPs represent an important special case, which, like the lexicographic ordering in utility theory, is worth studying for that reason alone. Within the CSP context, this is both because of its useful representational features and its amenability to ordinary CSP algorithms, in contrast to soft constraint representations.

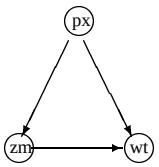
By assuming a mixture of hard and soft constraints instead of a single evaluation function, we are able to apply ordinary CSP algorithms or to combine them with strategies based on the preference ordering. In contrast, soft constraint representations require some form of branch and bound to solve to completion. (This is true even when hard constraints are represented within a soft constraints framework, as in [BFBW92] [SFV95].) It is to be expected that at least in certain parts of the problem space ordinary CSP algorithms will outperform branch and bound algorithms. (Of course, branch and bound techniques are also possible with ordinal CSPs, as shown later.)

Moreover, representation of a lexicographic ordering within a soft constraint framework (e.g. semiring or constraint hierarchy) will require an evaluation structure (or cost function) that is potentially unwieldy. This is because differences in evaluation between domain values for higher-priority variables must be larger than any possible sum of evaluations for assignments to lower-priority variables. In fact, an ordinal CSP can be readily transformed into a weighted CSP by using base arithmetic, with the base, b, equal to one greater than the largest domain size. Then, the kth value of the lowest-priority variable is associated with cost $k \times b^0$, the kth value on the next-lowest with $k \times b^1$, etc. (Given the "overwhelming advantage" of assignments to lexically prior variables, it might be thought that a fuzzy formulation is possible. But this ignores the fact that a the value associated with a particular assignment cannot be used in and of itself to determine the ordering of assignments.)

2.3 Ordinal CSPs and CP-nets.

Another formalism for representing certain kinds of preference structures is the conditional preference network with *ceteris paribus* assumptions, or CP-net [BBHP99]. Here, the preference ordering for one attribute depends on the value chosen for other attributes.

The CP-net formalism cannot represent lexicographic orderings (as noted by the authors). In addition, hard constraints cannot be construed as equivalent to conditional preferences inducing a directionality between variables of difference priority and altering preference orderings among values in the lower priority domain. Using the example of Section 1, this would give the following conditional preference graph and *ceteris paribus* conditions:



3: dig $\succ$ opt, 0.7 $\succ$ 0.5
2: dig $\succ$ opt, 0.5 $\succ$ 0.7
1: opt $\succ$ dig, 0.5 $\succ$ 0.7
dig: 0.5 $\succ$ 0.7
opt: 0.5 $\succ$ 0.7

However, given this cyclic structure, it is not possible to rank, for example, 3,dig,0.7 and 3,dig,0.5 using *ceteris paribus* assumptions. And, of course, this representation undercuts the distinction between preferences and hard constraints that is one of the strong points of the ordinal CSP formulation (e.g. the ordering 0.5 $\succ$ 0.7 given an optical zoom is based on an actual preference since an optical zoom can be combined with either weight, while the same ordering given a digital zoom is due to a constraint restriction).

A more recent extension of CP-nets, called "TCP-nets" [BD02], is capable of representing lexicographic orderings. However, even this framework does not allow one to clearly specify hard constraints as well as preference orderings, and so it does not allow one to use CSP algorithms in the straightforward fashion that ordinal CSPs do. Moreover, this formalism has a certain awkwardness in that it posits three kinds of edges in a graph, to represent importance and conditional importance as well as conditional preference.

Although the ordinal CSP framework does not allow one to express conditional preferences as can be done with CP-nets, it may be possible to combine the present framework with that of conditional CSPs (sometimes called "dynamic CSPs" [MF90]) to achieve the same representational capacities. The virtue of such an approach, in contrast to TCP-nets, is that it does not require any additional representational elements; instead it associates the usual nodes, arcs and values with more complex logical conditions. And since ordinary CSP algorithms can be extended for use with conditional CSPs, the study of such algorithms in the present context should also be relevant to this formalism.

3 Solving Ordinal CSPs with Ordinary CSP algorithms

Although it is usual to represent optimality with a cost function, which then allows solving by means of branch and bound, the peculiarites of ordinal CSPs make this approach problematic (shown below). In addition, the special features of these problems make them amenable in many cases to solving with ordinary CSP algorithms. In particular, if there are many feasible solutions, then employing a simple lexical ordering of variables and values may be all that is needed. On the other hand, when this is not reasonable, this will be because there are too few feasible solutions, and it is just this condition that may allow good CSP algorithms to be used, i.e. those designed to find feasible solutions quickly. For these reasons, we begin our discussion of solving methods with a study of ordinary CSP algorithms. In addition, since good heuristics should be transferrable to the branch and bound case, results relevant for such strategies can also be established.

Although it has the advantage of simplicity, a lexically ordered search disregards problem features that lead to rapid discovery of a solution, so it may be very slow on

many problems. On the other hand, if a non-lexical order is used, then the algorithm must solve the all-solutions problem in order to prove optimality. Therefore, although they will be more efficient in finding solutions when problems are hard (in CSP terms), it is still not clear how competitive these algorithms will be for solving ordinal CSPs.

To evaluate this issue, we compared performance of lexically-ordered search with that of two well-known variable and value ordering heuristics. The heuristics were (dynamic) minimum domain size variable ordering, where selection is based on the current sizes of the domains, and (dynamic) minimum-conflicts value ordering, where values with a smaller sum of conflicts with values in adjacent future domains are chosen before values with a larger number of conflicts [FD95]. (Although value ordering cannot make the all-solutions problem more efficient, it was of interest to determine whether such heuristics can be used to find good solutions more quickly.)

For experimental purposes, a total preference ordering 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. 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 labeling, is the most preferred, (1/1, 2/1, 3/1, ..., n/2) the next-most preferred, etc. And 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 number.

Random problems of this type, of course, have the same distribution of solutions as ordinary CSPs, so 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 [CKT91]. 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 may still see a crossover at some point between lexically-ordered search and search based on ordinary CSP heuristics.

Random problems were generated 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. Domain size was always 10. For the smallest number of variables, this allowed the entire plane of parameter values for density and tightness to be explored; however, for purposes of presentation representative values were chosen for density 0.5, so a slice of the plane can be shown that spans the entire range of tightness. Algorithms for forward checking (FC) [HE80] and maintained arc consistency using AC-3 (MAC-3) [SF94] were written in C, and experiments were run on a Dell Work Station PWS 330 with 4 CPUs running at 1800 MHz. The measures taken were time and numbers of backtracks, search nodes, and constraint checks. Since the four measures were reasonably well-correlated, number of nodes expanded is used to indicate search effort in the graphs below.

Figure 2 shows the search effort required by lexical ordering versus search with good CSP heuristics. As expected, lexical search ordering is superior when there are many solutions. In contrast, ordinary heuristics are grossly inefficient over this part of

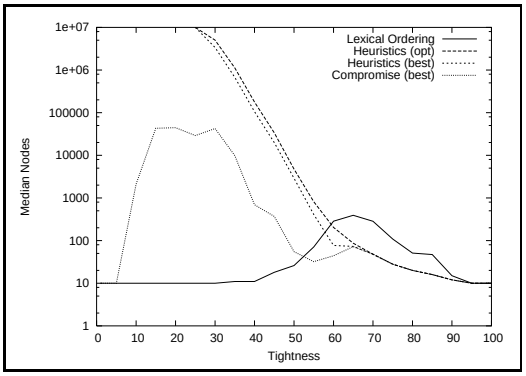


Fig. 2. Effort required, (i) to discover the best solution (labeled "best"), (ii) to prove it optimal (labeled "opt"). The algorithm used here was forward checking, combined with either lexical ordering or good heuristics for ordinary CSP. Ten-variable problems, with fixed density = 0.5 and tightness varying in steps of 0.05. Sample size at each step was 50 problems. For tightness ≥ 0.65 problems have no solutions. As an additional comparison, the effort required by a compromise ordering to discover the best solution is shown. (As one would expect, the "opt" curve for this strategy is almost identical to that for the good-heuristics algorithm.)

the parameter space. This applies to the effort required to find the best solution as well as that required to determine optimality. However, these results also show that, 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. Of importance is the fact that the crossover point occurs before (i.e. to the left of) the point where problems no longer have solutions and where lexical ordering must, of course, 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 on CSP heuristics (although it is not superior for determining optimal solutions). This suggests that it may be well-suited for either branch and bound or anytime procedures.

Use of MAC-3 produced greater savings for the lexical ordering without as much effect on the others, so that for these small problems the crossover was no longer apparent. However, with larger problems peak search effort grows progressively (Figure 3). And as problems get larger, the easy region also diminishes. Therefore, while MAC with lexical ordering is well-suited to small problems everywhere in the problem space, for larger problems it would still be useful to have more powerful techniques for some portions of the space.

The phase transition curve for lexical ordering shown in these figures does not represent quite the same phenomenon (or variant) as do complexity peaks with ordinary CSP or SAT algorithms [CKT91] [MSL92]. It is more closely related to the demonstra-

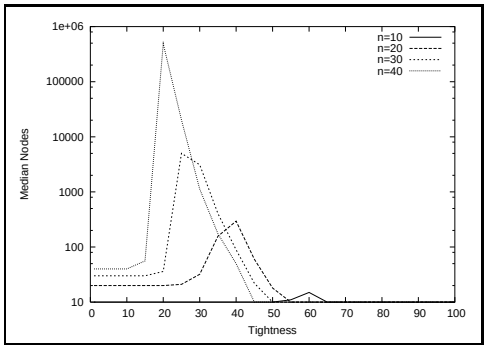


Fig. 3. Effort required by MAC-3 with lexical ordering, for 10, 20, 30 and 40-variable problems with density = 0.5 and tightness varying in steps of 0.05. Sample size at each step was 50 problems.

tion by [LM96] of an algorithm-dependent phase transition with increasing constraint tightness for MAX-CSPs. In their situation, as in the present case, problems can become more and more difficult across a parameter range, without showing a peak complexity, as it becomes progressively harder to discriminate the optimal from other assignments. At the same time, if an algorithm can either discriminate between these assignments, as lexical ordering does in the present case, or can discriminate *against* equivalent solutions via lower bounds, as the PFC-DAC algorithm does in the MAX-CSP case, then at some point problems become progressively easier, so that for these algorithms a phase transition is observed.

4 Branch and bound with CSP heuristics.

An alternative approach is to approach the optimization problem entailed by ordinal CSPs more directly, using a technique like branch and bound to find the best solution. In addition, we can easily 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 that uses base arithmetic, but 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. (Note that with dynamic variable ordering, when we are at level k in the search tree, we cannot assume that all $k-1$ lexically prior variables have been instantiated.)

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 the best assignment for that variable; 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.

Unfortunately, this procedure can be inefficient whenever variables of high priority occur late in the search order. This is shown in the following example, where there are 5 variables, and a solution has just been found using the following search order:

SEARCH LEVEL	VARIABLE ORDER	VALUE ASSIGNED
1	3	2
2	4	2
3	2	2
4	1	2
5	5	2

When we choose the next value for variable 5, we can only choose 1; otherwise we go over the bound. The same is true when we back up to level 4. But above this level, we cannot back up unless the uninstantianted variables of higher priority have the value 1 removed from their domains, since the cost function is dominated by these variables. In addition, lower bound calculations cannot be extended to priority levels below the current variable, since once a better value is chosen for the current variable, then any value can be chosen for variables with lower priority.

On the other hand, if the most-preferred variables are not placed too far down in the search order, a branch and bound algorithm may do appreciably better than ordering heuristics that do not take the preference ordering into account. In particular, when problems became very hard, with very few solutions, it may be possible to enhance ordinary CSP heuristics with a branch and bound strategy.

Therefore, we conjectured that the algorithm using lexical ordering would still find an optimal solution faster in most of the space. It was still possible, however, that the branch and bound algorithm would prove more efficient in a somewhat larger portion of the space than for ordinary, non-lexical CSP algorithms.

In the following tests, a branch and bound algorithm was used that also employed the “compromise” heuristics, i.e. minimum domain size variable ordering and lexical value ordering. In Figure 4, we see that our apprehensions about this procedure were unfortunately confirmed. Branch and bound does marginally better than the ordinary CSP heuristics, but there is still a large swath of the problem space where lexical ordering is markedly superior.

5 Anytime strategies.

There may be situations where an alternative approach is preferred to lexical ordering even when it does not finish as quickly. In particular, if the alternative tends to find the optimal solution faster than the lexical ordering on average, then we may prefer to use the former to obtain probably optimal solutions after a brief time. Given the total ordering, we can also determine how many steps any solution this algorithm finds is from the theoretical best solution, i.e. the optimal solution without any hard constraints, and this gives some idea of how quickly it approaches this optimum for a given class of problems. This may be sufficient for our purposes.

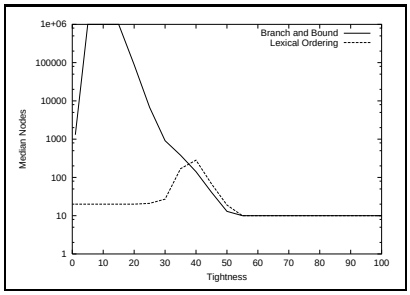


Fig. 4. Comparison of branch and bound for ordinal CSPs with lexical ordering. Branch and bound uses MAC and compromise heuristics, as described in the previous section. Twenty-variable problems, with density 0.5 and tightness varying in steps of 0.05.

To evaluate this contention, we plotted the "best solution found so far", for branch and bound as well as ordinary CSP algorithms. The distance measure is simply the cost function using a base of 10. We chose problems from a part of the problem space where lexical and non-lexical search are roughly comparable with respect to finding an optimal solution. For these problems, anytime curves show a rapid descent over the first hundred milliseconds for branch and bound and the compromise heuristics until they approximate the optimal solution, while for the variable/value heuristics the curve remains above the others for the first 2-3 sec (cf. Figure 5; similar curves were found for forward checking). It seems clear that some forms of non-lexical search have anytime properties that allow them to approximate optimal solutions quickly even for larger problems.

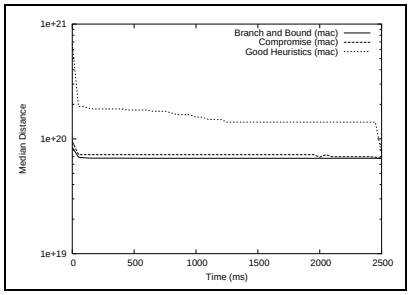


Fig. 5. Anytime curves for branch and bound with ordinary CSP algorithms, using MAC. Twenty-variable problems, density = 0.5, tightness = 0.25.

6 Preprocessing

If we can find a good solution (i.e. acceptable and high in the preference ordering) quickly with a specific preprocessing strategy, we may be able to reduce overall search effort. This section recounts the current results of our work-in-progress in this domain.

We tried dividing the space of alternatives so as to 'diagonalize' the problem to reflect the priority ordering of assignments. This means taking successively greater numbers of the highest priority values in domains of successively lower priority variables. For domain size = n, this means taking the first value for variable 1, the first two values for variable 2, ..., and all values for variable n. After the initial division, we used the rule: subtract from the top (and right) if a solution was found and add from the bottom (and left) otherwise. This continued until the amount to add or remove reached the value $\mid d \mid /n$. Unfortunately, given the overwhelming advantage factor it is impossible to keep the final partition for the subsequent search phase. However, if search does not find a better instantiation for the first k variables in the lexical ordering, then the next domain can be restricted to values at least as good as that found during preprocessing. To ensure completeness, the branch and bound procedure was used starting with the best solution found during preprocessing.

Some results with this strategy are shown in Figure 6. The measure of effort for the preprocessing strategy includes both the preprocessing phase and subsequent branch and bound. With this strategy, search effort is reduced to a modest degree in the hard region in comparison to lexical ordering, and there is some improvement over other non-lexical search strategies across the range of soluble problems.

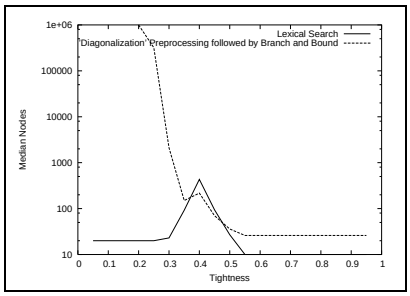


Fig. 6. Comparison of preprocessing followed by branch and bound search for ordinal CSPs with lexical ordering. Twenty-variable problems, with density 0.5 and tightness varying in steps of 0.05.

This approach can be refined by calculating the fraction of assignments left by the first diagonalization in relation to the expected number of solutions in the problem. (The latter can be estimated from the problem parameters.) For example, with 20 variables and 10 values per domain, the diagonalization described above leaves 10^{13} assignments out of 10^{20}, i.e. about one ten-millionth, while for tightness = 0.40, the expected number of solutions is 0.08. In this case, better results could probably be achieved by allowing

more values, either by 'shifting the diagonal to the right' or by adding values according to the rule described above. This possibility is currently under investigation.

7 Conclusions.

Ordinal CSPs offer a means of representing both preferences and hard constraints in a clear-cut fashion. At the same time, they allow straightforward use of powerful algorithms drawn from the CSP domain.

For small problems, a simple lexical ordering combined with the MAC algorithm can find optimal solutions very quickly. The same is true in the "easy range" for much larger problems. Standard CSP algorithms outperform lexical ordering over a limited portion of the hard-problem region, and the ability of some of these ("compromise" ordering and the branch and bound) to find an optimal solution quickly make them candidates for use as anytime algorithms. In addition, we have been able to develop algorithms that exploit the special features of ordinal CSPs to give further improvements in efficiency in some cases on harder problems.

Acknowledgements. We thank B. O'Sullivan, C. Domshlak and P. Snow for helpful comments on this material.

References

[BBHP99] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *Proc., 15th Ann. Conf. on Uncertainty in AI*, pages 71–80, San Mateo, CA, 1999. Morgan Kaufmann.

[BD02] R. I. Brafman and C. Domshlak. Tcp-nets for preference-based product configuration. In *ECAI02 Workshop on Configuration*, 2002.

[BDP97] S. Benferhat, D. Dubois, and H. Prade. Possibilistic and probabilistic semantics of conditional knowledge. In *Proc., 14th Nat. Conf. on AI*, pages 70–75, Menlo Park, CA/ Cambridge, MA, 1997. AAAI Press/ MIT.

[BFBW92] A. Borning, B. Freeman-Benson, and M. Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, 5(3):223–270, September 1992.

[BMR97] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint solving and optimization. *Journal of the ACM*, 44(2):201–236, March 1997.

[CKT91] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the *really* hard problems are. In *Proceedings, 12th Int. Jt. Conf. on AI*, pages 331–337, San Mateo, CA, 1991. Morgan Kaufmann.

[FD95] D. Frost and R. Dechter. Look-ahead value ordering for constraint satisfaction problems. In *Proc., 14th Int. Jt. Conf. on AI*, pages 572–578, San Mateo, CA, 1995. Morgan Kaufmann.

[FHK+02] A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Global constraints for lexicographic orderings. In P. Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP2002, LNCS 2470*, pages 93–108, Berlin, 2002. Springer.

[Fis68] P. Fishburn. Utility theory. *Management Science*, 14(5):335–378, January 1968.

[HE80] R. M. Haralick and G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.

[KR93] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives. Preferences and Value Tradeoffs*. Cambridge, New York, 1993.

[LM96] J. Larrosa and P. Meseguer. Phase transition in max-csp. In W. Wahlster, editor, *Proc., 12th Eur. Conf. on AI*, pages 190–194, Chichester, 1996. Wiley.

[Luc56] R. D. Luce. Semiring orders and a theory of utility discrimination. *Econometrica*, 24:178–191, 1956.

[MF90] S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *Proc., 8th Nat. Conf. on AI*, pages 25–32, Menlo Park/Cambridge, 1990. AAAI Press/MIT.

[MSL92] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of sat problems. In *Proc., 10th Nat. Conf. on AI*, pages 459–465, Menlo Park/Cambridge, 1992. AAAI Press/MIT.

[SF94] D. Sabin and E. Freuder. Contradicting conventional wisdom in constraint satisfaction. In A. Cohn, editor, *Proc., 11th European Conf. on AI*, pages 125–129, Chichester, 1994. John Wiley and Sons.

[SFV95] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proc., 14th Int. Jt. Conf. on AI*, pages 631–637, San Mateo, CA, 1995. Morgan Kaufmann.

[Sno94] P. Snow. The emergence of ordered belief from initial ignorance. In *Proc., 12th Nat. Conf. on AI*, pages 281–286, Menlo Park, CA/ Cambridge, MA, 1994. AAAI Press/ MIT.

[Tve72] A. Tversky. Elimination by aspects: A theory of choice. *Psychological Review*, 79:281–299, 1972.

[vWE86] D. von Winterfeldt and W. Edwards. *Decision Analysis and Behavioral Research*. Cambridge University Press, Cambridge, 1986.

[Wal96] R. J. Wallace. Analysis of heuristic methods for partial constraint satisfaction problems. In E. C. Freuder, editor, *Principles and Practice of Constraint Programming - CP'96*, volume 1118 of *Lecture Notes in Computer Science*, pages 482–496. Springer, Berlin, 1996.