

Exploiting Solution-Space Structure

David W. Etherington
CIRL
1269 University of Oregon
Eugene OR 97404
USA

ether 'at' cirل.uoregon.edu

Andrew J. Parkes
CIRL
1269 University of Oregon
Eugene OR 97404
USA

parkes 'at' cirل.uoregon.edu

ABSTRACT

We give a brief summary of work on exploiting solution space structure. The approach is to exploit structures that we call “solution clusters.” We describe what we mean by solution clusters, why they are relevant to reducing search within negotiation, and how they can be found. We give a high-level description of how they might be applied to a system-of-systems such as the SNAP-MAPLANT interaction.

1. INTRODUCTION

Suppose we have some combinatorial optimization or decision problem and an associated, exponentially large, search space. Typical examples include propositional satisfiability and scheduling. Such problems are often hard because the number of solutions is typically much less than the size of the search space. That is, solutions are very rare within the search space. However, earlier studies of problem instances from the phase transition region of random satisfiability problems [2], showed that, despite the rareness, the solutions still tend to cluster together. Thus, by a “solution cluster” we are simply referring to a cluster of solutions within the overall search space: that is, a region of the search space that is rich in solutions, despite their overall scarcity.

It seems to be natural that many problems will have solution clusters. A familiar example of this would be in standard scheduling problems. Suppose we demand that the schedule be optimal, in the sense of having the shortest possible overall length. Optimal schedules are very rare within the space of all schedules. However, despite this rarity, the optimal schedule is likely to be part of a solution cluster. To see this, consider the critical path within the schedule; these are the tasks whose start times cannot be perturbed without increasing the overall schedule length. Tasks that are not on the critical path can, by definition, be moved to some extent without breaking optimality. If the optimal solution were a single isolated solution then all paths would be on the critical path. In practice, this is very unlikely. An optimal solution is likely to be surrounded by a solution cluster corresponding to the results of moving tasks not on the critical path.

In the following we will explain why this is relevant to negotiating systems. In all cases we will take a high-level descriptive and motivational approach, and refer to the paper [3] for formal results, and further details. Unlike that paper, however, in this context we will be discussing the communication of solution clusters. It is essential that this does not lead to excessive communication costs; and so the set of solutions should have a compact representation. Hence, we adopt the following definition.

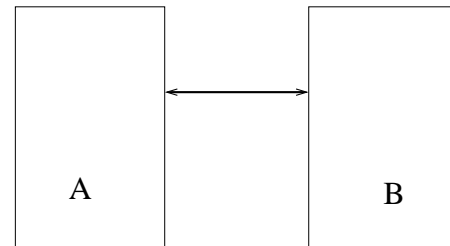


Figure 1: Two systems

Definition: A solution cluster is a region of the search space that:

- contains a large (exponential) number of solutions
- can be represented compactly; that is, has a polynomial size representation. Ideally, the representation should not be significantly larger than that for a single solution.

We emphasize that the “definition” is not meant formally, but merely in a descriptive sense. In practice, the exact form of the definition will need to be tailored to the domain and system requirements.

2. SOLUTION CLUSTERS, NEGOTIATIONS, AND AVOIDING DISTRACTIONS

Suppose that we have a “system-of-systems”. For simplicity we will consider just two systems, *A* and *B*, as in see Figure 1. A good example to keep in mind would be the SNAP system [?] interacting with the MAPLANT system [?].

Suppose that systems *A* and *B* are cooperating to try to find a good solution to a hard optimization/decision problem. Both *A* and *B* have access to their portion of the overall problem, but neither can see the entire problem. Since the portions interact, the systems cannot find solutions alone, but must ensure that their solutions are compatible with the other system’s. Typically, the search within a single system will involve constructing solutions: working from partial assignments (of values to variables) and trying to build a complete solution. Optionally, it might also involve attempts to find solution clusters (we will describe how to do this later). At any stage it could communicate its partial results to the other agent(s). However, the act of communication is expensive. Besides the transmission costs, there is also the possibility of distracting the other agent from its own work.

The general issue here is that of deciding how much work to do before communicating your (partial) results to others. In human terms, at one end of the spectrum, one might communicate even the slightest progress to others, but this might result in only result in distracting them from work they need to be doing, with little positive effect. At the other end of the spectrum, one could work privately until having what seems to be a perfect solution, based on the data you know, only to discover very quickly some data that clearly invalidates the entire proposed solution.

In terms of search, the issue is which one of the following to use for communications:

1. Partial solutions. In the extreme case, every time the partial assignment is extended and does not result in internal failure (that is, it seems fine internally), then it could be sent to the other agent as a potential portion of a solution.
2. Single solutions. Once the system has a complete satisfying assignment then it is transmitted.
3. "Solution clusters". Even after finding a single solution, the system could continue to work and convert it into a solution cluster before transmitting.

(In reality, these are just the natural points within a whole spectrum of choices; for example, one could decide to just transmit whatever solution one had at some regular time interval.)

All of these have their pros and cons, as follows:

1. Partial solutions.

Pros **Early Pruning.** The advantage of early communication is that bad regions of the search space can be pruned early. System A might spend a lot of effort on a partial assignment P , even though B might know that P is fatally flawed and should be immediately discarded. However, part of the system design should be to partition the overall problem in such a way as to reduce the chances of this happening.

Cons **Distraction** of other agents. If the problem is well-partitioned then most partial assignments will not be pruned by the other agents. In this case, the frequent transmission of partial assignments will merely distract the other agent from its own search problem, and not improve the search. In an extreme case the changes inside one agent might correspond to fixing values within some very easy sub-problem, and so be essentially irrelevant to the other agent(s).

2. Single solutions. This is the standard choice.

Pros Once we have a local solution, it is natural and simple to try to extend it to a global solution by sending it to other agents.

Cons It is possible that the single solution from A will be incompatible with B's solution due to many "trivial" reasons. The resulting negotiations to fix these "merging defects" might be straightforward, but they still might correspond to many rounds of negotiations, and associated increase in communications costs.

3. Solution Clusters

Pros The intended advantage of sending a solution cluster is that it is an inexpensive way to send an exponential

number of solutions to the other agent. The other agent can then select one that is most compatible with its own solution. Since it has, effectively, an exponential number of choices, the hope is that the chance of defects is much reduced. The paper [3] provides some experimental support for this hoped-for advantage.

Cons After all the effort to find a solution cluster, another agent might tell that there is a simple defect that invalidates the entire cluster, and so corresponds to the loss of all the associated computational effort. For this reason, it is important that the computational cost of finding a cluster not be too high.

Note that the notion of distraction is well-known in the community, (see, for example, [1]) but has been limited to cases of partial assignments and single solutions. The work at CIRL allowed the spectrum of communication choices to be increased, and allows even lower distraction by permitting communications to be delayed until a solution cluster is available.

3. SOLUTION CLUSTERS IN SAT

Given the potential advantages of solution clusters, the immediate questions are:

- how can one more precisely define clusters?
- how can one find clusters?
- are their advantages actually realized?

The paper [3] answers these questions for the case that each system is a satisfiability problem, SAT. Brief summaries of the results follow

3.1 Defining solution clusters for SAT

In SAT we define a solution cluster in terms of partial assignment P (that is, a set of values for a subset of all the variables). The residual theory from P is just the set of constraints that are not satisfied by P , that is, the constraints that remain between the variables not assigned a value by P .

For P to correspond to a solution cluster, we simply require that the residual theory is underconstrained. That is, it should not only be satisfiable, but (informally) have many solutions, and they should be very easy to find. In SAT, this corresponds to the residual theory having a low clause to variable ratio. Optionally, we might also require that the residual theory not force any residual variable to a single value, otherwise this forced value should simply be added to P .

The actual definition is complicated somewhat by the fact that not all variables in a system might be visible to the other systems. Variables can be split into private and public, and the relevant definition of a solution cluster should refer only to the public variables.

3.2 Finding Solution Clusters

The simplest algorithm is a bottom up greedy search. It starts with a single solution found using some standard search procedure, and then iteratively tries to "relax" to a solution cluster by unsetting variables, trying to select variables that allow the largest cluster, and stopping when the residual theory is about to become too constrained:

PROCEDURE 3.1 (FIND-SOLUTION-CLUSTER). *Given a SAT problem C , to compute a partial assignment solution cluster:*

```

1 Find a single solution,  $P \leftarrow \text{SOLVE}(C)$ 
   else return failure
2 while ( $\text{resid}(P)$  is under-constrained )
3    $P' = P$ 
4   select variable  $x \in P$  such that  $\text{resid}(P-x)$  is least constrained
5    $P = P-x$ 
6 return  $P'$  (the latest under-constrained  $P$ )

```

Detecting the constrainedness of $\text{resid}(P-x)$ is usually just a matter of counting unsatisfied clauses, and hence is relatively fast. If necessary, the variables x that are tested can be restricted to come from some small, heuristically selected set, rendering the algorithm even faster.

Also, there is no attempt here to find a maximum sized cluster, the greedy search simply finds some cluster that is locally maximally sized. It is important to note that the proposed usage of solution clusters does not in any way require that they be optimal.

Again this was modified somewhat by the need to consider the public and private variables, but the essential idea, a bottom-up greedy relaxation, still applies.

3.3 Experimental Results on Solution Clusters

We considered a simple scenario of multiple systems trying, in a single round of negotiations, to find a global solution. The success metric was simply the probability of success. Results showed that, in the interesting cases in which the problem was neither too easy nor insoluble, that passing solution clusters drastically reduced the probability of failure of the negotiations.

4. RELEVANCE TO SNAP-MAPLANT

The existing work [3] only considered the case in which the constraints are represented as a satisfiability problem. In the real world constraints are likely to have a much more complicated representation, however, we expect that the essential idea “find a single solution and then greedily relax it to solution cluster” will still be applicable.

In the case of MAPLANT, the constraints are solved by use of the constraint-programming (CP) system, Mozart, hence here we sketch out how such an approach might be used in a CP system. In implementing the algorithm in CP we can of course use the existing search within CP in order to find the initial system. In CP there is usually no direct support for unsetting a variable, however, this is presumably straightforward to implement by building up a separate computation based on the partial assignment, $P - x$, to be tested.

In CP, the most difficult might be to implement a measure of the constrainedness of the residual theory. Presumably, it is possible to detect which constraints are already satisfied by P . It is tempting to then just keep a count of the unsatisfied constraints. If all the constraints are primitive (built-in, simple, constraints) then this is reasonable. However, in CP, one makes extensive use of global constraints (such as the “cumulative constraint” that is typically used to enforce resource bounds in scheduling problems). These involve many, if not all, of the variables of the problem, and it is unlikely that they will be totally satisfied. It is likely that methods will need to be developed to determine whether the residual global constraints are loosely constrained or not. However, such information is likely to also be of use in building heuristics for the initial search; for example, it is common to try to select the least constraining values, and so measurements of “constrainedness” are likely to be of general use.

Also, in CP, it is usual to use a finite-domain encoding rather than a boolean encoding. In this case a natural definition of a so-

lution cluster would be to view the partial assignment as a set of unary constraints on variables, and instead of only allowing equality constraints, to also allow inequalities. For example, P might be $\{x \leq 3, y = 8, y \geq 2\}$. The consequences of such unary constraints will be determined by the inbuilt propagation methods in CP systems, and the resulting ranges on variables will correspond to the residual theory.

5. CONCLUSIONS

Here we gave an overview of the theory of solution clusters, their relevance to negotiation, methods to find them, and potential applicability to systems of systems such as SNAP-MAPLANT.

6. ACKNOWLEDGMENTS

This was funded by DARPA under contract number ????????

7. REFERENCES

- [1] D. Mammen and V. Lesser. Problem structure and subproblem sharing in multi-agent systems. In *International Conference on Multi Agent Systems (ICMAS 98)*, 1998. citeseer.nj.nec.com/mammen98problem.html.
- [2] A. J. Parkes. Clustering at the Phase Transition. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 340–345, Providence, RI, 1997.
- [3] A. J. Parkes. Exploiting solution clusters for coarse-grained distributed search. In *Proceedings of the workshop on “Distributed Constraint Reasoning”, held at “Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)”*, August 2001.