Borrett, J.E., Tsang, E.P.K. & Walsh, N.R.,
Adaptive Constraint Satisfaction,
in Proc., Workshop of the
UK Planning and Scheduling
Special Interest Group,
Liverpool, UK, 1996

Adaptive Constraint Satisfaction

J. E. Borrett, E. P. K. Tsang & N. R. Walsh

Dept. of Computer Science
University of Essex
Wivenhoe Park
Colchester CO4 3SQ
United Kingdom
email: {jborrett, edward, walsnq}@essex.ac.uk

August 1996

Abstract; Many different approaches have been applied to constraint satisfaction. These range from complete backtracking algorithms to sophisticated distributed configurations. However, most research effort in the field of constraint satisfaction algorithms has concentrated on the use of a single algorithm for solving all problems. At the same time, a consensus appears to have developed to the effect that it is unlikely that any single algorithm is always the best choice for all classes of problem.
In this paper we argue that an adaptive approach should play an important part in constraint satisfaction. This approach relaxes the commitment to using a single algorithm once search commences. As a result, we claim that it is possible to undertake a more focused approach to problem solving, allowing for the correction of bad algorithm choices and for capitalising on opportunities for gain by dynamically changing to more suitable candidates.

1. INTRODUCTION

The recent growth in research in the field of constraint satisfaction has resulted in a large range of different approaches to solving constraint satisfaction problems. Overviews of some of these techniques can be found in (Tsang 93) and (Kumar 92). More recently, combination of complete and incomplete methods has become popular (Richards&Richards 96).

Conventional CSP solving systems such as CHIP, ECLiPSe and ILOG Solver tend to operate by using a single search technique regardless of the problem presented to it[1]. This approach means that the problem solver is forced into making the commitment to a fixed search technique at the outset of the problem solving process. Since the science of selecting the most appropriate algorithm for a given problem is a an outstanding problem in the domain of constraint satisfaction, the process of choosing an algorithm is clearly error prone. As a result, the use of a “generally good” algorithms is

[1] We acknowledge that some of these packages allow for some flexibility in defining some elements of the search algorithm such as the search ordering.

usually adopted. However, this approach is sub-optimal since in an ideal world it would always be desirable to have the most appropriate algorithm in use as our solver.

Given that there is such a wide range of potentially useful problem solving methods, how does a problem solver choose the most appropriate technique to solve their problem? This question has been addressed in (Tsang&Kwan 94) (Tsang et al 95) where the mapping of algorithms to different problems is considered. In (Tsang et al 95) evidence is presented showing that clear domains exist where different algorithms are more suited to different classes of CSP.

One approach to overcoming the problem of algorithm selection is to apply learning techniques over a set of training instances of problems. This was the approach adopted in the MultiTAC system (Minton 93). MultiTAC takes a set of typical problem instances and constructs a suitable algorithm using learning techniques. It was shown to have potential constructing algorithms, using a set of predefined algorithm elements, for a restricted class of problems.

Another possible method for improving the selection of an algorithm is to generate decision trees as can be seen in (Kwan 97). Here, algorithms are classified for specific classes of CSP using fuzzy logic techniques, based on training sets of typical problems. Once again, this technique is shown to be effective in improving on the use of a single algorithm.

While the techniques mentioned above for improving algorithm selection suggest that we can improve on the use of a generally robust algorithm, they still exhibit a major drawback which comes from the fact that they rely on a statistical measures such as the average performance. The problem with this is revealed dramatically in work on the so-called exceptionally hard problems (Smith 94) (Hogg&Williams 94) (Gent&Walsh 94) (Smith&Grant 95). These works demonstrate that there are sometimes dramatically high search cost instances of problems for some algorithms, in a given class of CSPs. The observation that there is sometimes a significant distribution of “difficulties” in a class of CSPs means that the identification of suitable algorithms for particular classes of CSP cannot translate to totally accurate mappings.

The observations outlined above present a clear opportunity for gain by using adaptive constraint satisfaction techniques. If techniques can be developed such that we can determine how a particular algorithm is performing, we then have the information with which to take corrective action in the event that an incorrect initial selection of algorithm has been made. Such an approach relies on having a suite of candidate algorithms available for this corrective action, but could for example allow for the use of less robust yet faster algorithms, safe in the knowledge that if that particular algorithm is performing ineffectively, we can switch to another choice.

A second opportunity for gain lies with the modification or change to an initial search algorithm when the possibility for improvements in the current performance are observed. For example there is a high diversity in problem structure, there may be the potential for using different algorithms to solve different parts of that problem.

In order to exploit the opportunities outlined above, for more effective CSP solving, we believe there is a requirement for some form of adaptive qualities in the problem solving technique. We have previously defined the notion of adaptive constraint satisfaction (ACS) in (Borrett et al 96). There we describe it as;

Adaptive Constraint Satisfaction is a general philosophy for solving constraint satisfaction problems. It aims to make use of the many algorithms and techniques available by relaxing the *commitment to a single algorithm when solving a particular CSP, allowing for the active modification or switching of algorithms during the search process.*

In the remainder of this paper we attempt to give a general overview of our adaptive approach to constraint satisfaction. In the next section we describe a general model which we have developed to facilitate the development of adaptive techniques. In section 3 we give details of particular instances of ACS. Finally, in section 4 we present a discussion of the future directions for ACS developments.

2. A GENERAL MODEL FOR ADAPTIVE CONSTRAINT SATISFACTION (ACS)

The impetus behind the idea of adaptive constraint satisfaction arises form the desire to make use of the many and varied techniques available for use in constraint satisfaction. As hinted at in the definition of section 1, the key to any such approach is the ability to make changes to the solving technique during the search. Effectively there are three main components which are needed to facilitate this. The first of them is the set of algorithms which we intend to make available. There are clearly many alternatives for this, given the large range of algorithms available. The next element is to provide a source of information with which informed decisions about the progress of search can be made. We describe such sources of information as *monitors*. The third and final element is the overall guiding *strategy* which makes the decisions about if, when and how to change the current method for searching.

These basic elements, the algorithms, the monitors and the strategy are illustrated in figure 1. This shows how it is the strategy that binds the adaptive solver together.

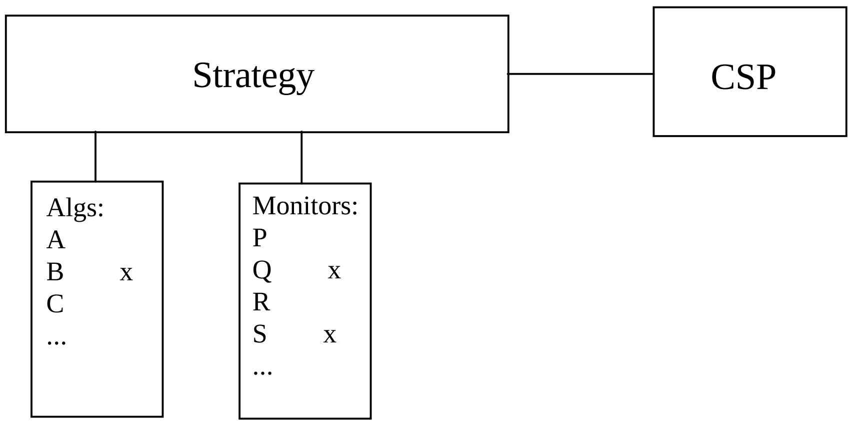

Figure 1 - A monitor-based approach to Adaptive Constraint Satisfaction

In that figure 1, an “x” adjacent to an algorithm or monitor indicates that they are currently active. Under different situations, different algorithms and monitors may be switched on.

2.1. Adaptive Strategies

The main role of an adaptive strategy is to act as a guide to the selection of appropriate algorithms at appropriate times, according to the information available to it. An example is to use a *restart* strategy. Here the algorithms used by the strategy are invoked to completely restart the search once information from the monitors suggests that the current selection is not performing effectively. This type of approach might be used in situations where we want to ensure that bad performance of an algorithm is terminated, cutting our losses, starting afresh with a new algorithm which is less likely to succumb to the same problem. Similarly this approach can also be used opportunistically if it is clear from information gained during the search that a different algorithm would be better, even if the current algorithm is not performing badly.

Another type of strategy is a *constructive* strategy, where any information about the search carried out so far is maintained and used after a change in working algorithm takes place. This is a sensible strategy, especially in situations where only the variable or value heuristic is to be changed and the algorithm is to remain unchanged. This type of strategy can be seen as constructively building on any partial solution that has been generated up to that point. Again there is the possibility for switching out of an algorithm which is performing badly, or for opportunistic change to a potentially better algorithm.

There are clearly many possible strategies which can be developed. However, they all have certain common features. More specifically, all strategies have a selection of algorithms from which they can choose when any change in the state of solving is detected. Strategies also have a selection of monitors to use for monitoring the performance of the algorithms in use. What is needed from a strategy is clear efficiency improvement on the traditional approach of using a single algorithm.

2.2. Monitors

As we have shown above, monitors are an essential part of any adaptive solving technique since they provide the information which strategies need to function effectively. Without good monitors, adaptive strategies would cease to function. Similarly, more intelligent monitors give the potential for more innovative strategy designs. The importance of good monitors should not be under stated.

Monitors can be viewed in different ways. One way is to view them as looking for problems with the progress and performance of an algorithm, such as thrashing, the type behaviour which can be seen in backtracking algorithms. Another type of monitor is the opportunistic type which looks out for potential improvements on the current approach to searching, hence directing a strategy to a potentially more effective way of solving than the current one.

Many candidates exist for useful monitors. These can either be algorithm specific, or applicable to a more general class of algorithm. Some examples include;

- The execution time used
- Levels of pruning in the case of lookahead algorithms
- The level of backtracking in the case of backtracking algorithms
- Analysis of search space characteristics at certain points in the search

Of course there may be many more candidates, far too many to list here. However, the main point is that monitors are a key part of any adaptive strategy. For a monitor to be a useful one it should be cheap to evaluate, thus minimising the overheads incurred by the strategy. It is clear that monitors should also provide useful information to strategy, and coupled to this, monitors should provide accurate information to the strategy if effective decisions are to be made. The accuracy will determine the frequency at which incorrect strategic decisions are made, which is something that must be minimised.

3. EXAMPLES OF ACS

The idea of using of adaptive techniques to solve constraint satisfaction problems is a relatively new one. For that reason there are few examples of its use. However, in this section we present two examples. One is a strategy specifically designed to overcome the problem of exceptionally hard problems (Borrett et al 96) and the other is an example of how algorithms can be fine tuned during the search process. Both examples demonstrate the potential of adaptive techniques.

3.1 REBA

REBA is a strategy that was designed to overcome the problem of exceptionally hard problems (Smith 94). It is based on the use of a chain of algorithms which define a fixed order of usage and these are applied in a "restart" fashion whereby no information is carried over to successive algorithms. A single monitor is used to provide the strategy with information, thus indicating when a switch is needed. In the sections below we present an outline of how REBA was designed.

3.1.1 Chain Design

At the heart of the strategy used by REBA is a chain of algorithms. These algorithms are chosen to cover a range of sophistication, but at the same time there is a realisation that less sophisticated algorithms tend to be more susceptible to EHPs. This idea is illustrated in table 1.

Rank	Algorithm Sophistication	Median Cost	Sensitivity to EHPs
1	X	X	Z
2	Y	Y	Y
3	Z	Z	X

Table 1. How the ranking of algorithms can differ when based on median cost of solving CSPs, and sensitivity to EHPs.

If we can determine rankings similar to those in Table 1, we would have enough information to design a useful chain for solving CSPs in the easy region whilst increasing the likelihood of avoiding the potentially catastrophic effects of encountering an EHP. The chain can simply be set to an ordering of algorithms based on the "Quickest First Principle" (QFP), where quickest indicates the algorithm with the best median performance.

While there is some overhead involved in this approach, the benefits can be considerable. For example, the ability to use a simple algorithm can result in an order of magnitude gain in performance over its more complex counterparts. Another advantage is that in the event of a bad initial choice of algorithm, we are not stuck with it. Mistakes of this nature will be rectified when we switch away from the bad choice.

The chain used by REBA consists of a selection of algorithms with good median performance on low density, easy soluble CSPs, and a selection of algorithms with good worst case performance. Having carried out some preliminary investigations, we chose to use the following algorithms;

BM+MWO - back-marking (Gaschnig 77) with the minimum width ordering (Freuder 82)

BMCBJ+MWO - back-marking with conflict-directed backjumping (Prosser 93) and the minimum width ordering

BMCBJ+MDO - back-marking with conflict-directed backjumping and the maximum degree ordering[2]

FCCBJ+BZ - forward checking with conflict-directed backjumping (Prosser 93) and the Brélaz ordering (Brélaz 79)

MAC+MDO - maintain arc consistency (Sabin&Freuder 94) with the maximum degree ordering

We chained these algorithms in the following way;

$$\text{BM+MWO} \rightarrow \text{BMCBJ+MWO} \rightarrow \text{BMCBJ+MDO} \rightarrow \\ \text{FCCBJ+BZ} \rightarrow \text{MAC+MDO}$$

The reasoning behind this chain is that BM+MWO is very fast for many easy soluble problems, but very susceptible to thrashing. However, it might succeed in finding a solution quickly, otherwise thrashing will be detected. In the event that BM+MWO fails, we try adding intelligent backjumping to it. If this fails, we try changing the ordering, since a bad ordering is often a contributing factor to EHPs. If these simpler algorithms fall victim to an EHP, we attempt to use forward checking with conflict-directed backjumping and a dynamic variable ordering. Finally, if this fails, we resort to another algorithm which has relatively low susceptibility to EHPs, MAC+MDO.

3.1.2 Switching policy

Switching in REBA is governed by the information provided by a single monitor. The aim of this monitor is to predict the likelihood of thrashing in the current algorithm. This monitoring activity is achieved using the Monitor Search Level, MSL, (Borrett et al 96) thrashing predictor. MSL attempts to predict when thrashing type behaviour is likely to occur such that only a small portion of any futile sub-search space is actually explored by the algorithm in question.

One indication of thrashing is when the search from a particular level *i* never proceeds beyond a certain depth, *d*, and that a large proportion of the search space between level *i* and level *i*+*d* is explored (i.e. little pruning takes place between these two levels). Such a situation can occur when the culprits of the failure at level *i*+*d* precede the level *i*. MSL is a simple method which uses this observation to predict thrashing type behaviour by observing patterns of the behaviour of backtracking levels during the search process. For full details of how it operates and evaluation of its effectiveness we refer the reader to (Borrett et al 96).

[2] A static ordering based on descending order of degree.

3.1.3 Effectiveness of REBA

The work in (Borrett et al 96) has shown that REBA can be an effective strategy for solving CSPs in the so-called easy region. Work is in progress to evaluate the usefulness of REBA type strategies on other classes of CSP.

3.2 Adaptive Propagation

Constraint propagation is a technique used by many commercial constraint solving packages. It aims to reduce the size of the sub-space below a given node by eliminating values from the domains of unlabelled variables. When values are committed to at certain nodes in the search tree, it is possible to achieve such a reduction by checking the consistency of values in the remaining future variables, using the constraints. Various algorithms have been developed in order to enforce different level of consistency. Examples of these are forward checking (Haralick&Elliott 80), arc-consistency (Mackworth 77) and path-consistency (Mackworth 77).

In (Sakkout et al 96) the possibility of varying the amount of propagation at different points in the search space is considered. In particular, they consider the possibility or using either FC or AC as a basis for propagation. The idea here is that AC incurs more effort to enforce, but has the benefit of providing increased levels of domain reduction when compared to FC, in some cases. However, this is not always the case and some times AC provides no advantage over FC while incurring extra cost. The result of this approach has been an algorithm called Adaptive Arc Propagation (AAP).

AAP fits the model of adaptive algorithms since it combines two different approaches to solve the problem and uses information about the search in order to change its method of solving. The use of FC or AC is co-ordinated using a monitor which analyses the effects of any problem reduction which takes place. Experiments presented in (Sakkout et al 96) suggest that this is a promising line of research.

4. DISCUSSION

The idea of adding adaptive qualities to constraint satisfaction problem solving methods has been outlined in this paper. It is a new approach and presents the research community with the opportunity for making greater use of the many new and varied CSP solving algorithms, based on the use of strategies and monitors. The fact that domains for algorithms have been demonstrated makes this an attractive proposition.

The two examples presented in this paper have demonstrated that adaptive techniques can be successfully applied to CSP solving and that significant benefits can result. A further important advantage of using adaptive strategies is that they reduce the significance of algorithm selection by the problem solver. This is possible since the adaptive approach can rectify any incorrect decision that has been made.

In fact, we believe that monitors should naturally accompany algorithms as they are developed, for the following reason: it is the responsibility of researchers who propose new algorithms to provide methods for evaluating their effectiveness. For example, one may measure the number of values which are deleted by lookahead algorithms, or the number of variables jumped over in a back-jumping algorithms. Such measures can then be used to design monitors for the corresponding algorithms allowing the community to make use of them more easily in future adaptive systems.

Acknowledgements

This work was supported by EPSRC research grant ref. GR/J/42878.

References

Borrett, J. E., Tsang, E. P. K. & Walsh, N. R, Adaptive Constraint Satisfaction: The Quickest First Principle, Proceedings 12th European Conference on Artificial Intelligence, 160-164, 1996

Brélaz, D., New methods to color the vertices of graphs, Communications of the ACM, 22(4), 251-256, 1979

Freuder, E. C., A sufficient Condition for Backtrack-Free Search, Journal of ACM, vol. 29, 24-32, 1982

Gaschnig, J., A General Backtrack Algorithm That Eliminates Most Redundant Tests, Proceedings 5th International Joint Conference on Artificial Intelligence, 457, 1977

Gent, I. P. & Walsh, T., Easy Problems are Sometimes Hard, Artificial Intelligence **70**, 335-345, 1994

Haralick, R. M. & Elliott, G. L., Increasing Tree Search Efficiency for Constraint Satisfaction Problems, Artificial Intelligence **14**, 263-31, 1980

Hogg, T. & Williams, C., The hardest Constraint Problems: A double Phase Transition, Artificial Intelligence **69**, 155-171, 1994

Kwan, A. C. M., PhD thesis, to appear 1997.

Kumar, V., Algorithms for Constraint Satisfaction Problems: A Survey, AI Magazine, 13(1), 1992, 32-44

Mackworth, A. K., Consistency in Networks of Relations, Artificial Intelligence, Vol. 8, 99-118, 1977

Minton, S., An Analytical Learning System for Specializing Heuristics, Proceedings 13th International Joint Conference on Artificial Intelligence, 922-928, 1993

Prosser, P., Hybrid Algorithms for the Constraint Satisfaction Problem, Computational Intelligence, Vol. 9, 268-299, 1993

Richards, E. T. & Richards, B., Nogood Learning for Constraint Satisfaction, Proceedings, Second International Conference in Principles and Practice of Constraint Programming (CP'96), August, 1996, to appear

Sabin, D. & Freuder, E. C., Contradicting Conventional Wisdom in Constraint Satisfaction, Proceedings 11th European Conference on Artificial Intelligence, 24-129, 1994

Sakkout, H. E., Wallace, M. G. & Richards, E. B., An Instance of Adaptive Constraint Propagation, Proceedings, Second International Conference in Principles and Practice of Constraint Programming (CP'96), August, 1996, to appear

Smith, B., In Search of Exceptionally Difficult Constraint Satisfaction Problems, Proceedings of the Workshop on Constraint Processing, 11th European Conference on Artificial Intelligence, 79-86, 1994

Smith, B. & Grant, A., Sparse Constraint Graphs and Exceptionally Hard Problems, Proceedings 14th International Joint Conference on Artificial Intelligence, 646-651, 1995

Tsang, E. P. K., 1993, Foundations of Constraint Satisfaction, Academic Press, London

Tsang, E. P. K., Borrett, J. E. & Kwan A. C. M., 1995, An Attempt to Map a Range of Constraint Satisfaction Algorithms and Heuristics, in Proceedings AISB 95, 203-216.

Tsang, E. P. K. & Kwan A. C. M., 1993, Mapping Constraint Satisfaction Problems to Algorithms and Heuristics, Technical Report CSM254, Dept. of Computer Science, University of Essex, Colchester