View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by NASA Technical Reports Server

137254

N93-18687

An Extended Abstract: A Heuristic Repair Method for Constraint-Satisfaction and Scheduling Problems

Steven Minton[1] and Mark D. Johnston[2] and Andrew B. Philips[1] and Philip Laird[3]

[1]Sterling Federal Systems
NASA Ames Research Center
AI Research Branch
Mail Stop: 269-2
Moffett Field, CA 94035 USA

[2]Space Telescope Science Institute
3700 San Martin Drive,
Baltimore, MD 21218 USA

[3]NASA Ames Research Center
AI Research Branch
Mail Stop: 269-2
Moffett Field, CA 94035 USA

Introduction

One of the most promising general approaches for solving combinatorial search problems is to generate an initial, suboptimal solution and then to apply local *repair* heuristics. Techniques based on this approach have met with empirical success on many combinatorial problems, including the traveling salesman and graph partitioning problems[10]. Such techniques also have a long tradition in AI, most notably in problem-solving systems that operate by debugging initial solutions [18, 20]. In this paper, we describe how this idea can be extended to constraint satisfaction problems (CSPs) in a natural manner (see also [14] for full paper).

Most of the previous work on CSP algorithms has assumed a standard backtracking approach in which a partial assignment to the variables is incrementally extended. In contrast, our method starts with a complete, but inconsistent assignment and then incrementally repairs constraint violations until a consistent assignment is achieved. The method is guided by a simple ordering heuristic for repairing constraint violations: identify a variable that is currently in conflict and select a new value that minimizes the number of outstanding constraint violations.

We present empirical evidence showing that on some standard problems our approach is considerably more efficient than traditional backtracking methods. For example, on the *n*-queens problem, our method quickly finds solutions to the one million queens problem[15]. We argue that the reason that repair-based methods can outperform backtracking methods is because a complete assignment can be more informative in guiding search than a partial assignment. However, the utility of the extra information is domain dependent.

The work described in this paper was inspired by a surprisingly effective neural network developed by Adorf and Johnston [1] for scheduling astronomical observations on the Hubble Space Telescope. Our heuristic CSP method was distilled from an analysis of the network. In the process of carrying out the analysis, we discovered that the effectiveness of the network has little to do with its connectionist implementation. Furthermore, the ideas employed in the network can be implemented very efficiently within a symbolic CSP framework. The symbolic implementation is extremely simple. It also has the advantage that several different search strategies can be employed, although we have found that hill-climbing methods are particularly well-suited for the applications that we have investigated.

We begin the paper with a brief review of Adorf and Johnston's neural network. Following this, we describe our symbolic method for heuristic repair.

Previous Work: The GDS Network

By almost any measure, the Hubble Space Telescope scheduling problem is a complex task [11, 17]. Between ten thousand and thirty thousand astronomical observations per year must be scheduled, subject to a great variety of constraints including power restrictions, observation priorities, time-dependent orbital characteristics, movement of astronomical bodies, stray light sources, etc. Because the telescope is an extremely valuable resource with a limited lifetime, efficient scheduling is a critical concern. An initial scheduling system, developed using traditional programming methods, highlighted the difficulty of the problem; it was estimated that it would take over three weeks for the system to schedule one week of observations. This problem was remedied by the development of a successful constraint-based system to augment the initial system. At the heart of the constraint-based system is a neural network developed by Adorf and Johnston, the Guarded Discrete Stochastic (GDS) network, which searches for a schedule[1].

From a computational point of view, the network is interesting because Adorf and Johnston found that it performs well on a variety of tasks, in addition to the space telescope scheduling problem. For example, the network performed significantly better on the *n*-queens problem than methods that had been previously developed. The *n*-queens problem requires placing *n* queens on an $n \times n$ chessboard so that no two queens share a row, column or diagonal. The network has been used to solve problems of up to 1024 queens, whereas most heuristic backtracking methods encounter difficulties with problems one-tenth that size[19].

The GDS network is a modified Hopfield network[8]. In a standard Hopfield network, all connections between neurons are symmetric. In the GDS network, the main network is coupled asymmetrically to an auxiliary network of *guard neurons* which restricts the configurations that the network can assume. This modification enables the network to rapidly find a solution for many problems, even when it is simulated on a serial machine. Unfortunately, convergence to a stable configuration is no longer guaranteed. Thus the network can fall into a local minimum involving a group of unstable states among which it will oscillate. In practice, however, if the network fails to converge after some number of neuron state transitions, it can simply be stopped and started over.

To solve the n-queens problem with the GDS network, each of the $n \times n$ board positions is represented by a neuron whose output is either one or zero depending on whether or not a queen is located in that position. (Note that this is a local representation rather than a distributed representation of the board.) If two board positions are inconsistent, then an inhibiting connection exists between the corresponding two neurons. For example, all the neurons in a column will inhibit each other, representing the constraint that two queens cannot be in the same column. For each row, a guard neuron is connected to each of the neurons in the row and gives the neurons in that row a large excitatory input, large enough so that at least one neuron in the row will turn on. Thus, the guard neurons enforce the constraint that one queen in each row must be on. The network is updated on each cycle by randomly picking a row and flipping the state of the neuron in that row whose input is most inconsistent with its current output. A solution is realized when the output of every neuron is consistent with its input.

Why does the GDS Net Perform So Well?

Our analysis of the GDS network was motivated by the question: "Why does the network perform so much better than traditional backtracking methods on certain tasks?" In particular, we were intrigued by the results on the n-queens problem, since this problem has received considerable attention from previous researchers. For n-queens, Adorf and Johnston found empirically that the network requires a linear number of transitions to converge. Since each transition requires linear time, the expected (empirical) time for the network to find a solution is $O(n^2)$. To check this behavior, Johnston and Adorf ran experiments with n as high as 1024, at which point memory limitations became a problem.[1]

[1]The network, which is programmed in Lisp, requires approximately 11 minutes to solve the 1024 queens problem on a TI Explorer II. For larger problems, memory becomes a limiting factor because the the network requires approximately $O(n^2)$ space.

Nonsystematic Search Hypothesis

Initially, we hypothesized that it was the nonsystematic nature of the network's search that allowed it to perform much better than systematic depth-first backtracking search. There are two potential problems associated with systematic depth-first search. First, the search space may be organized in such a way that poorer choices are explored first at each branch point. For instance, in the n-queens problem, depth-first search tends to find a solution much more quickly when the first queen is placed in the center of the first row rather than the corner. It would appear that solution density is much greater in the former case[19], but most naive algorithms tend to start in the corner simply because humans find it more natural to program that way. However, the fact that a systematic algorithm may consistantly make poor choices does not completely explain why the GDS network performs so well for n-queens. A backtracking program that randomly orders rows (and columns within rows) performs much better than the naive method, and yet still performs poorly relative to the GDS network.

The second potential problem with depth-first search is more significant and more subtle. Depth-first search can be a disadvantage when solutions are not evenly distributed throughout the search space. As the distribution of solutions becomes less uniform, and, therefore, the solutions become more clustered, the time to search between solution clusters increases. Thus, we conclude that, in a tree where the solutions are clustered, depth-first search performs relatively poorly. In comparison, a search strategy which examines the leaves of the tree in random order is not affected by solution clustering.

We investigated whether this phenomenon explained the relatively poor performance of depth-first search on n-queens by experimenting with a randomized search algorithm, called a Las Vegas algorithm [2]. The algorithm begins by selecting a path from the root to a leaf. To select a path, the algorithm starts at the root node and chooses one of its children with equal probability. This process continues recursively until a leaf is encountered. If the leaf is a solution the algorithm terminates, if not, it starts over again at the root and selects a path. The same path may be examined more than once, since no memory is maintained between successive trials.

The Las Vegas algorithm does, in fact, perform better than simple depth-first search on n-queens. In fact, this result was already known [2]. However, the performance of the Las Vegas algorithm is still not nearly as good as that of the GDS network, and so we concluded that the systematicity hypothesis alone cannot explain the network's behavior.

Informedness Hypothesis

Our second hypothesis was that the network's search process uses information about the current assignment that is not available to a standard backtracking program. We now believe this hypothesis is correct, in that it explains why the network works so well. In particular, the key to the network's performance appears to be that state transitions are made so as to reduce the number of outstanding inconsistencies in the network; specifically, each state transition involves flipping the neuron whose output is most inconsistent with its current input. From a constraint satisfaction perspective, it is as if the network reassigns a value for a variable by choosing the value that violates the fewest constraints. This idea is captured by the following heuristic:

Min-Conflicts heuristic:

Given: A set of variables, a set of binary constraints, and an assignment specifying a value for each variable. Two variables *conflict* if their values violate a constraint.

Procedure: Select a variable that is in conflict, and assign it a value that minimizes the number of conflicts.[2] (Break ties randomly.)

We have found that the network's behavior can be approximated by a symbolic system that uses the min-conflicts heuristic for hill-climbing. The hill-climbing system starts with an initial assignment generated in a preprocessing phase.[3] At each choice point, the heuristic chooses a variable that is currently in conflict and reassigns its value, until a solution is found. The system thus searches the space of possible assignments, favoring assignments with fewer total conflicts. Of course, the hill-climbing system can become "stuck" in a local maximum, in the same way that the network may become "stuck" in a local minimum.

There are two aspects of the min-conflicts hill-climbing method that distinguish it from standard backtracking approaches for CSP problems. First, instead of extending a consistent partial assignment, the min-conflicts method *repairs* a complete but inconsistent assignment by reducing those inconsistencies. Thus, to guide its search, it uses information about the current assignment that is not available to a standard backtracking algorithm. Second, the use of a hill-climbing strategy produces a different style of search.

We have also found that extracting the method from the network enables us to tease apart and experiment with its different components. In particular, the idea of repairing an inconsistent assignment can be used with a variety of different search strategies in addition to hill-climbing.

[2]In general, the heuristic attempts to minimize the number of other variables that will need to be repaired. For binary CSPs, this corresponds to minimizing the number of conflicting variables. For general CSPs, where a single constraint may involve several variables, the exact method of counting the number of variables that will need to be repaired depends on the particular constraint. The space telescope scheduling problem is a general CSP, whereas the other tasks described in this paper are binary CSPs.

[3]See [14] for an analysis of how different initial assignments can affect the repair phase.

Highlights of Experimental Results

This section contains highlights from experiments in which we evaluate the performance of the min-conflicts heuristic on some standard tasks. These experiments identify problems on which min-conflicts performs well, as well as problems on which it performs poorly. The experiments also show the extent to which the min-conflicts approach approximates the behavior of the GDS network.

The N-Queens Problem

- Min-conflicts hill-climbing approximates the GDS network for n-queens.
- For $n \geq 100$ min-conflicts hill-climbing has never failed to find a solution.
- For min-conficts, the required number of repairs appears to remain *constant* as n increases, and the time to find a solution grows linearly with n.
- Standard backtracking using the "most-constrained first" heuristic quickly grows large: for 100 runs when $n > 1000$ a backtracking program implementing the heuristic took more than 12 hours to complete.
- Min-conflicts hill-climbing solves the million queens problem in less than four minutes on a SPARCstation I.
- N-queens is actually quite an easy problem given the right method.

Scheduling Applications: HST

- Min-conflicts hill-climbing approximates the GDS network for HST scheduling.
- Much of the overhead (particularly the space overhead) in the GDS network is eliminated by using the min-conflicts method.
- Because the min-conflicts heuristic is so simple, a min-conflicts scheduler for HST was quickly coded in C and is extremely efficient.
- The simplicity of the min-conflicts method makes it easy to experiment with modifications to the heuristic and the search-strategy.
- Other telescope scheduling problems have started to use the min-conflicts scheduler developed for HST.

Graph Coloring

- Min-conflicts hill-climbing approximates the GDS network for graph coloring.
- A standard backtracking algorithm employing a Brelaz-like[3] heuristic outperforms min-conflicts hill-climbing.

Summary of Experimental Results

For each of the three tasks we have examined in detail, n-queens, HST scheduling and graph 3-colorability, we have found that the GDS network's behavior can be approximated by the min-conflicts hill-climbing algorithm. To this extent, we have a theory that explains the network's behavior. Obviously, there are certain practical advantages to having "extracted" this method from the network. First, the method is very simple, and so can be programmed extremely efficiently, especially if done in a task-specific manner. Second, the heuristic we have identified, that is, choosing the repair which minimizes the number of conflicts, is very general. It can be used in combination with different search strategies and task-specific heuristics, an important factor for most practical applications.

Insofar as the power of our approach is concerned, our experimental results are encouraging. We have identified two tasks, n-queens and HST scheduling, which appear more amenable to our repair-based approach than a traditional approach that incrementally extends a partial assignment. This is not to say that a repair-based approach will do better than *any* traditional approach for solving these tasks, but merely that our simple, repair-based method has done relatively well in comparison to the standard traditional methods. We also note that repair-based methods have a special advantage for scheduling tasks: they can easily be used for both overconstrained and rescheduling problems. Thus it seems likely that there are other applications for which our approach will prove useful.

Discussion

The heuristic method described in this paper can be characterized as a *local search* method[10], in that each repair minimizes the number of conflicts for an individual variable. Local search methods have been applied to a variety of important problems, often with impressive results. For example, the Kernighan-Lin method, perhaps the most successful algorithm for solving graph-partitioning problems, repeatedly improves a partitioning by swapping the two vertices that yield the greatest cost differential. The much-publicized simulated annealing method can also be characterized as a form of local search[9]. However, it is well-known that the effectiveness of local search methods depends greatly on the particular task.

In fact, it is easy to imagine problems on which the min-conflicts heuristic will fail. The heuristic is poorly suited for problems with a few highly critical constraints and a large number of less important constraints. For example, consider the problem of constructing a four-year course schedule for a university student. We may have an initial schedule which satisfies almost all of the constraints, except that a course scheduled for the first year is not actually offered that year. If this course is a prerequisite for subsequent courses, then many significant changes to the schedule may be required before it is fixed. In general, if repairing a constraint violation requires completely revising the current assignment, then the min-conflicts heuristic will offer little guidance.

The problems investigated in this paper, especially the HST and n-queens problem, tend to be relatively uniform in that critical constraints rarely occur. In part, this is due to the way the problems are represented. For example, in the HST problem, as described earlier, the transitive closure of temporal constraints is explicitly represented. Thus, a single "after" relation can be transformed into a set of "after" relations. This improves performance because the min-conflicts heuristic is less likely to violate a set of constraints than a single constraint. In some cases, we expect that more sophisticated techniques will be necessary to identify critical constraints[5]. To this end, we are currently evaluating explanation-based learning techniques [4, 13] as a method for identifying critical constraints.

The algorithms described in this paper also have an important relation to previous work in AI. In particular, there is a long history of AI programs that use repair or debugging strategies to solve problems, primarily in the areas of planning and design[18, 20]. This approach has recently had a renaissance with the emergence of case-based[6] and analogical [12, 21] problem solving. To solve a problem, a case-based system will retrieve the solution from a previous, similar problem and repair the old solution so that it solves the new problem.

The fact that the min-conflicts approach performs well on n-queens, a well-studied, "standard" constraint-satisfaction problem, suggests that AI repair-based approaches may be more generally useful than previously thought. However, in some cases it can be more time-consuming to repair a solution than to construct a new one from scratch.

There are many possible extensions to the work reported here, but three are particularly worth mentioning. First, we expect that there are other applications for which the min-conflicts approach will prove useful. Conjunctive matching, for example, is an area where preliminary results appear promising. This is particularly true for matching problems that require only that a good partial-match be computed. Second, we expect that there are interesting ways in which the min-conflicts heuristic could be combined with other heuristics. Finally, there is the possibility of employing the min-conflicts heuristic with other search techniques. In this paper, we considered only one very basic method, hill climbing. However, since the number of conflicts in an assignment can serve as a heuristic evaluation function, more sophisticated techniques such as best-first search are possible candidates for investigation. Another possibility is Tabu search[7], a hill-climbing technique that maintains a list of forbidden moves in order to avoid cycles. Morris[16] has also proposed a hill-climbing method which can break out of local maxima by systematically altering the cost function. The work by Morris and much of the work on Tabu search bears a close relation to our approach.

Conclusions

In this paper we have analyzed a very successful neural network algorithm and shown that an extremely simple, heuristic search method behaves similarly. Based on our experience with both the GDS network and min-conflicts hill-climbing, we conclude that the min-conflicts heuristic captures the critical aspects of the GDS network. In this sense, we have explained why the network is so effective. Additionally, by isolating the min-conflicts heuristic from the search strategy, we distinguished the idea of a repair-based CSP method from the particular strategy employed to search within the space of repairs.

Finally, there are several practical implications of this work. First, the scheduling system for the Hubble Space Telescope, SPIKE, now employs our symbolic method, rather than the network, reducing the overhead necessary to arrive at a schedule. Second, and perhaps more importantly, it is easy to experiment with variations of the symbolic method, which should facilitate transferring SPIKE to other scheduling applications. Third, by demonstrating that repair-based methods are applicable to standard constraint satisfaction problems, such as N-queens, we have provided a new tool for solving CSP problems.

References

[1] H.M. Adorf and M.D. Johnston. A discrete stochastic neural network algorithm for constraint satisfaction problems. In *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, 1990.

[2] G. Brassard and P. Bratley. *Algorithmics - Theory and Practice.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[3] D. Brelaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22:251–256, 1979.

[4] M. Eskey and M. Zweben. Learning search control for constraint-based scheduling. In *Proceedings AAAI-90*, Boston, Mass, 1990.

[5] M.S. Fox, N. Sadeh, and C. Baykan. Constrained heuristic search. In *Proceedings IJCAI-89*, Detroit, MI, 1989.

[6] K.J. Hammond. *Case-based Planning: An Integrated Theory of Planning, Learning and Memory.* PhD thesis, Yale University, Department of Computer Science, 1986.

[7] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39:345–351, 1987.

[8] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, volume 79, 1982.

[9] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, Part II. *To appear in Journal of Operations Research*, 1990.

[10] D.S. Johnson, C.H. Papadimitrou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[11] M.D. Johnston. Automated telescope scheduling. In *Proceedings of the Symposium on Coordination of Observational Projects.* Cambridge University Press, 1987.

[12] S. Kambhampati. Supporting flexible plan reuse. In Minton S., editor, *Machine Learning Methods for Planning and Scheduling.* Morgan Kaufmann, 1992.

[13] S. Minton. Empirical results concerning the utility of explanation-based learning. In *Proceedings AAAI-88*, Minneapolis, MN, 1988.

[14] S. Minton, M. Johnston, A.B. Philips, and P. Laird. Minimizing conflicts: A heuristic repair method for constraint-satisfaction and scheduling problems. Submitted to *Special Issue of Artificial Intelligence Journal on Constraint Satisfaction Problems.*

[15] S. Minton, M. Johnston, A.B. Philips, and P. Laird. Solving large scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings AAAI-90*, 1990.

[16] P. Morris. Solutions without exhaustive search: An iterative descent method for binary constraint satisfaction problems. In *Proceedings the AAAI-90 Workshop on Constraint-Directed Reasoning*, Boston, MA, 1990.

[17] N. Muscettola, S.F. Smith, G. Amiri, and D. Pathak. Generating space telescope observation schedules. Technical Report CMU-RI-TR-89-28, Carnegie Mellon University, Robotics Institute, 1989.

[18] R.G. Simmons. A theory of debugging plans and interpretations. In *Proceedings AAAI-88*, Minneapolis, MN, 1988.

[19] H.S. Stone and J.M. Stone. Efficient search techniques - an empirical study of the n-queens problem. *IBM Journal of Research and Development*, 31:464–474, 1987.

[20] G. J. Sussman. *A Computer Model of Skill Acquisition.* American Elsevier, New York, 1975.

[21] M.M. Veloso and J.G. Carbonell. Towards scaling up machine learning: A case study with derivation analogy in prodigy. In Minton S., editor, *Machine Learning Methods for Planning and Scheduling.* Morgan Kaufmann, 1992.