Improving Search through Diversity

Peter Shell
Center for Machine Translation
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
pshell@cmu.edu

Juan Antonio Hernandez Rubio
Gonzalo Quiroga Barro
Union Fenosa
Capitán Haya, 53
28020 Madrid (Spain)
gquiroga@uef.es

Abstract

Adding diversity to symbolic search techniques has not been explored in artificial intelligence. Adding a diversity criterion provides us with a powerful new mechanism for finding global maxima in complex search spaces and helps to alleviate the problem of premature convergence to local maxima. A theoretical analysis is presented of issues in diversity searching which previously haven't been addressed, and a domain-independent diversity-search algorithm for practical breadth-first searching is developed. Empirical results of an implementation in the CRESUS expert system for intelligent cash-management confirm that diversity can significantly improve the solution quality of symbolic searchers.

Introduction

This paper presents a new symbolic technique for tackling a fundamental problem found in all search techniques used in complex domains: converging to a local maximum at the expense of not finding a global optimum. In large and complex search problems where it is impossible to visit a significant fraction of the possible states, heuristic evaluation functions must be used to decide which intermediate solutions to reject and which to pursue. This can cause pruning of locally suboptimal intermediate states even though they may have led to a better global solution. This recurring problem has not been studied in depth and thus there are not many good solutions to it.

This problem manifests itself in all heuristic search paradigms. In genetic algorithms (Holland 1975) one of the main problems affecting search performance is "premature convergence", where most of the members of the "population" end up being very similar to each other and many potentially useful "genes" are lost. This results in suboptimal solutions when the best solution included one of the lost genes. A similar problem occurs in breadth-first search as we will see.

In depth-first search such as game-trees, there is a similar problem called the "horizon effect". The evaluation function gives an inaccurate estimation of the cost of a state because it was at an unstable local maximum or minimum (for example threatening the king in chess), thereby misleading the search to prefer that state even though it may not ultimately lead to an optimum.

This paper develops and evaluates a method I call "diversity search". Diversity search maintains the uniqueness of intermediate states and operators in order to increase the coverage of the space searched. It promises to improve the performance of many types of searchers by alleviating the local-minimum problem. Although it can augment most search techniques, it has received little attention in artificial intelligence. The idea has been used in genetic algorithms (Mauldin 1984) and simulated annealing (Kirkpatrick, Gelatt & Vecchi 1983) but hasn't been attempted in symbolic search methods such as depth-first or breadth-first search. This paper reports work which has combined ideas found in genetic algorithms and simulated annealing with symbolic search techniques. I address some issues surrounding diversity search which haven't previously been considered, then develop a domain-independent algorithm for diversity-search in a breadth-first searcher. An implementation is presented in CRESUS (Shell et al. 1992), a working expert-system in the complex domain of cash-management, with empirical results that show that diversity search can significantly improve global search performance.

Related Work

Here we consider related solutions to the local maximum problem in different search formalisms and contrast them to diversity search. Although diversity search hasn't been addressed in symbolic search, it has been addressed in genetic algorithms. Genetic algorithms simulate natural evolution to optimize an evaluation function. They are analogous to breadth-search where the intermediate states are called "genomes"; sets of states correspond to populations; and the evaluation function corresponds to the physical environment. New states are generated by applying genetic operators such as crossover or mutation to single or pairs of parent genomes. Genomes with the highest evaluation function have the best probability of "reproducing." Search proceeds by reproducing and replacing genomes for typically several thousands of generations, and picking the most fit genomes from the resulting population. Genetic algorithms are mostly used in machine learning (DeJong 1990).

DeJong (DeJong 1975) recognized early on that premature convergence in genetic algorithms adversely affected performance and attempted to alleviate it. He employed probabilistic reproduction and operator mutation so that the locally favored genes wouldn't dominate the pool. Although this increased the diversity of the population it didn't significantly improve performance. Adding a "crowding factor", which limited the number of genotypes similar to the favored genotype, helped his system on his hardest test case (F5). Mauldin (Mauldin 1984) later introduced an explicit diversity criterion into genetic search and showed that diversity significantly improved performance using DeJong's 5 polynomial test functions. Booker (Booker 1987) developed several new crossover strategies (such as using two crossover points and strategically varying the crossover rate) which gave similar improvement without explicitly introducing diversity.

For several reasons, diversity in genetic algorithms doesn't transfer easily to symbolic search methods. Genetic algorithms usually require several thousand iterations to find an optimized solution. This works well for the simple polynomial functions used in much of the genetic-algorithms literature where the evaluation is fast and states are represented in few bits, but isn't feasible in knowledge-rich systems where the evaluation function takes longer to compute and the states are more complex. Furthermore, the two orthogonal criteria of diversity and primary evaluation cannot be combined in the same way by symbolic searchers, as we will see. Finally, the metric used to evaluate diversity in genetic algorithms is simply the "Hamming distance" between the bit-strings representing each genome. Since symbolic search methods don't usually use a bit-string representation this metric is not transferrable.

Simulated annealing (Kirkpatrick, Gelatt & Vecchi 1983) is a stochastic search technique designed to find minimum-cost solutions to large optimization problems without converging to a local minimum. It is based on an analogy with statistical mechanics. It has been used in such tasks as graph bisection (Jerrum & Sorkin 1993) and connectionistic networks such as the Boltzman machine (Ackley, Hinton & Sejnowski 1985) However, simulated annealing is a slow process which requires fast evaluation and generation functions (Davis & Steenstrup 1987) and so wouldn't be feasible for most knowledge-based expert-systems. Furthermore, as Mauldin (Mauldin 1984) pointed out, it is not intrinsically parallel like genetic algorithms.

Techniques have been developed for alleviating a similar problem in depth-first search. Quiescence search (Beal 1990), used largely in game programs to handle the horizon effect, identifies local maxima with respect to the (sometimes inaccurate) evaluation function by situations of instability such as checking a king in chess. Such situations are likely to "fool" the evaluation function into giving an inaccurate cost. When they occur the algorithm continues searching until a stable, or quiescent, state is reached. Depth-first searchers can't incorporate the diversity of states into their algorithms because they only retain one state at a time, and they need to compare the states to determine their diversity.

Berliner (private communication) has experimented with increasing the diversity of operators in depth-first searchers by grouping operators by similarity and ensuring that several different groups of operators are generated at each step in the search. This may encourage diversity at the operator-generation level but lacks the ability to ensure diversity among states.

Motivation

The introduction described the need for a technique to avoid convergence to a local maximum in order to find the global maximum, and noted the lack of research into diversity search in symbolic domains. Here we examine the behavior of a specific breadth-first searcher which will motivate introducing diversity into searching techniques.

Our breadth-first searcher is part of CRESUS (Shell et al. 1992), a cash-management expert-system used by the treasury department of Union Fenosa, an electric company serving greater Madrid in Spain. This expert system could be used by any medium or large-sized company to automate and improve their treasury. The task of the treasurer is to manage the daily cash-flow and cover the company's financial needs by borrowing money from some subset of the numerous company credit lines, balance several bank accounts, allocate payments and collections and invest any excess funds. Each such operation involves commissions and interest payments which vary depending on the amount of money and the associated financial instrument (e.g., check or wire), credit-line or bank-account. The goal is to minimize the global cost of all of these operations.

An artificial intelligence approach was developed after it was realized that non-heuristic algorithms such as linear programming couldn't find a solution in a reasonable amount of time due to the large number of variables in the problem. In the CRESUS searcher, an *operator* is a cash-management operation such as moving $1000 from account A to account B using a wire transfer. A *state* is defined by the current balance of all the bank accounts and amounts available in the credit lines. The evaluation function computes the total cost of the current operations plus the estimated cost of the remaining operations. A *solution* is a sequence of operations which balance all of the bank accounts. CRESUS is implemented in Common-Lisp and currently runs on Sun Sparcstations. It uses the PARMENIDES (Shell & Carbonell 1988) frame language as its knowledge representation and FRULEKIT (Shell & Carbonell 1986) for the inference engine.

An experienced treasurer requires several hours to manually complete the task. The searcher automates the decision-making and performs it in about 5 minutes with a hybrid beam-search (Newell 1978) algorithm. Further, the searcher usually outperforms the treasurer, finding solutions which cost about 20% less than the treasurer's.

The cash-management task is extremely complex. For each state in the search space, about 100 operators may apply, and solutions typically consist of a sequence of about 100 operators. Thus the search space contains about 10^{200} states. Furthermore, it is multi-modal; of high dimensionality and has a high degree of "epistasis": one part of a solution can be inhibited or modified by another part. This space is made tractable by encoding expert knowledge into the operator evaluation and generation functions, and by partitioning it into minimally interacting subspaces. These subspaces correspond to the daily solution versus the global, or period, solution of 2 to 4 weeks and the space of operators of a particular type.

Although the beam-searcher usually finds solutions which the human expert can't improve, occasionally it constructs suboptimal solutions corresponding to local maxima. It is important to avoid such solutions because the treasury department has started to rely on the searcher on a daily basis; and since the searcher usually finds good solutions, the expectations of the users have been raised. Furthermore, the company plans to market the system and would like it to perform consistently well.

Examining the states at which the searcher arrived at the end of the search illustrates the problem. The final states are extremely similar to each other. Table 1 shows the average number of operators constituting each state which are unique to that state, i.e. which don't appear in any other state. For example, if state *1* were created by the application of operators *{A,B,C}* and state 2 by operators *{B,D,A}* then those states would each have two unique operators, *C* and *D*, respectively (the computation of operator uniqueness is discussed in more detail in the next section). The beam-searcher was run over 5 days in October 1993 with a beam-width of 20. Similar results are obtained when the intermediate states are examined, for example after 10 or 20 iterations.

Table 1: Average number of unique and common operators in the beam searcher states

Day	1	2	3	4	5	Average
# unique ops	*1.7*	*1*	*1*	*1*	*2*	*1.34*
Operators in common	*95.9%*	*97.8%*	*97.1%*	*98.3%*	*96.7%*	*97.2%*

Intuition explains the high similarity between the states. When any type of searcher tries to find an optimal state in a large search space it will only be able consider a tiny subset of those nodes. For example a beam searcher might keep a set of on the order of the 20 best states and generate 10 potential successors to each state. If the average depth of a space is 100, then the searcher will only have considered $20 \times 10 \times 100$ or 20,000 states. Because the number of states visited by the searcher is so small compared to the number in the space (10^4 versus 10^{200}), intuition tells us that the states which have the highest evaluation will end up being very similar to each other. This is because the states which only differ from the preferred state by one operator or by a small change in an operator, will likely have one of the highest evaluations as well. This isn't due to an inaccurate evaluation function because the evaluation function had been finely tuned – it performs miniature searches of its own to accurately estimate the cost of the remaining operations. These observations led me to develop the diversity search algorithm.

The Diversity-search Algorithm

The heuristic of diversity search is that by increasing the "diversity" of the intermediate states in a breadth-first searcher, we may be able to improve the final results. By retaining some states not only because they have a desirable primary evaluation (such as cost) but also because they are sufficiently different from the other states, we can avoid converging on local maxima which can mislead the global searcher. A formalization of the standard beam-search algorithm is presented, followed by the diversity-based algorithm. Next two issues in diversity searching which haven't been sufficiently considered are addressed: how to define diversity of states and how to combine the primary evaluation with the diversity evaluation.

The Standard Beam-search Algorithm

To compare diversity-search with a more traditional algorithm and to introduce notation, the standard beam-search algorithm used in CRESUS is first presented.

a. Initialize set-of-states to a set containing one state which is the current problem's state.

b. REPEAT until no more states can be expanded:

   1. Expand set-of-states using generation function **G**.

   2. Evaluate the expanded states using primary evaluation function **E**.

   3. Pick the K highest-evaluated states as the next set of states.

The generation and evaluation functions are as follows:

G: $State_0 \Rightarrow \{State_1, State_2, \ldots State_n\}$

$State_i$ is derived by applying an operator to $State_0$.

E: $State \Rightarrow Evaluation$

Evaluation is the estimated cost of the given state. K, the beam width, is the number of states retained at each step.

The Diversity Beam-search Algorithm

The diversity beam-search algorithm augments steps b.2 and b.3 to include an evaluation of the diversity of states as well as primary evaluation:

a. *Initialize set-of-states to a set containing only the current problem's state.*

b. *REPEAT until no more states can be expanded:*

1. *Expand set-of-states using generation function* **G**.

2. *Evaluate the expanded states using primary evaluation function* **E**.

2'. *Evaluate the diversity of the expanded states using diversity evaluation function* **D**.

3. *Pick the K highest-evaluated states into the next set of states.*

3'. *Add the K' states which maximize the function* **C(E,D)** *to the next set of states.*

Where:

D: $State \Rightarrow Diversity\text{-}evaluation$

where *Diversity-evaluation* is the estimated diversity of the given state relative to the other states and:

C: $Cost\text{-}eval, Diversity\text{-}eval \Rightarrow Combined\text{-}eval$

K' is the diversity beam width, i.e., the number of states to keep at each step based on their diversity in addition to the K cost-wise best states.

The diversity-computation function **D** and combination function **C** are discussed next.

Computing the Diversity

Because diversity search hasn't previously been attempted in symbolic search techniques, the issue of computing the diversity of states hasn't been adequately addressed. An intuitive method would be to compare states directly using domain-specific functions. For example, in the cash-management domain, one might compare how many bank accounts are balanced and how much money is available in each credit-line. However, this requires domain-specific comparison functions and doesn't take into account that different sequences of operators with different costs could have led to the same state.

A more accurate way to compare states is to count how many constituent operators – independent of order – that they have in common, since it is the constituent operators which comprise the solution. In symbolic searchers where operators are applied to generate new states, the sequence of operators leading to each state can be associated with each state. To calculate the closeness between two states S_1 and S_2, we will define *State-match*(S_1,S_2) to be:

$$\sum_{i=1}^{Len(S_1)} \text{Op-state-match}\,(\text{Op-num}\,(i, S_1), S_2) \qquad \textbf{(EQ 1)}$$

where *Op-state-match(Op,S)* is defined as:

$$\text{Max}(\text{Op-match}(\text{Op},\text{Op-num}\,(i, S))\;)\;\; \forall i < Len\,(S) \qquad \textbf{(EQ 2)}$$

where *Op-num(i,S)* returns the i^{th} operator in state *S*.

Note that if the search technique compared states at different depths, such as A*, *State-match* would have to return a percentage instead of a count of matches.

The function *Op-match* should return a number 0 through 1, where 0 means that the operators are completely different and 1 means that they are identical. To simplify and to make the comparison domain-independent, we can define *Op-match* to return 1 only if the operators are identical and to otherwise return 0. Under this definition, *State-match*(S_1,S_2) tells us how many operators associated with S_1 are also associated with S_2.

Finally, to compute the degree of match of a state relative to a set of states, a logical method would be to compute the maximum match between the state in question and the rest of the states in the state-set. Thus *Match-in-Set(S,Set)* is:

$$\text{Max}(\text{State-match}(\text{S},\text{State-num}(\text{i},\text{Set}))\;)\;\; \forall i < Len(S) \qquad \textbf{(EQ 3)}$$

State-num(i,Set) gives the i^{th} state in the set of states Set.

The *diversity* of a state is then the number of operators in the state minus the *Match-in-Set* of that state.

Combining Primary and Diversity Evaluation

A harder question is how to combine the orthogonal criteria of primary cost-evaluation and diversity evaluation. We want to retain some states because of their diversity, but we should also retain some of the best cost-evaluated states so that the search proceeds towards the global minimum. In genetic algorithms, Mauldin (Mauldin 1984) limited the population size by requiring that all intermediate states have a minimum diversity from the others, and

probabilistically "reproduced" the state with the highest cost-evaluation. However, these two steps are merged in beam search and there is no analog to "reproduction" in symbol search, so a different technique is needed.

A simple method would be to maintain two different sets of states: one for the states which minimize the cost and one for the states which maximize the diversity function. This would cause combination function C to simply return the diversity and ignore the cost-evaluation. The heuristic is that if a state from the diversity-set becomes qualified it would "jump" to the cost-evaluation set.

A disadvantage of the "simple" approach is that the states in the diversity-set are not selected for cost-evaluation and so may stray towards extremely diverse but high-cost solutions. Thus a more sophisticated algorithm was developed which maintains multiple "bands" of diversity sets, each containing states within a given range of diversities. Within each band, the lowest-cost states are retained. Since states only compete with other states within the same band, high-diversity but higher cost states won't replace medium-diversity but lower cost states. Note that diversity is always computed by comparing to all states, not just the ones sharing the same band. The idea is that some of the high-diversity states will turn into medium-diversity states which in turn may eventually jump into the cost-evaluation set.

The parameters to the band searcher are the number and size of the bands and the minimum diversity. For example, if the minimum diversity were 2, the number of bands were 3 and the band-size were 2 then the first band would contain states with diversity 2 and 3, the second would contain states with diversity 4 and 5, and so on. The next section measures the effect of these diversity techniques on the global searcher.

Empirical Results

The standard CRESUS beam-searcher and 3 variations of the diversity searcher were run on 4 separate weeks of 1993 company treasury data. The total cost for each weekly solution is computed by summing the costs of credit-line dispositions, funds-movement and payment and collection commissions, and overdrawn-account fees. For the standard beam-searcher, the beam-width K of 10 was used; for the diversity searchers, the cost-eval beam-width K was 5 and the diversity-eval K' was 5. In this way we can determine if the 5 additional states are most effectively used as cost-eval states or diversity-eval states.

The Y axis in Figure 1 plots the cumulative total cost of solutions found by each search strategy. The simple diversity searcher, *simple*, and two variants of band-searching are shown. In the first variant, *band(4,1)*, there were 4 bands and the band-size was 1; in *band(3,2)*, the number of bands was 3 and the band-size was 2. The minimum diversity in both band searchers was 2.

All three diversity algorithms did better than standard beam-searching. Week 1 saw the largest improvement. It contained many local maxima because it contained a holiday and there were more constraints on funds-movements. In the standard search all of the states converged to a situation where idle funds were stuck in unusable accounts. In *simple* and *band(3,2)*, some funds had still been allocated to accounts which were affected by the holiday; in *band(4,1)*, all money had been correctly allocated. *Simple* also did surprisingly well on week 2 but worse than standard search on week 4. *Band(4,1)* did the best overall because of consistently good performance (see Table 2). *Band(4,1)* did better than *band(3,2)* because the latter suffers from the same problem as *simple*: within each band of size 2, slightly lower-cost states will be selected over more diverse states. Thus bands should probably always be of size 1. The reason that *band(4,1)* did slightly worse than *simple* on week 4 may be that that week didn't contain many local maxima which fooled *simple;* or that the local maxima fooled the diversity approaches as well as *simple*. In this case the benefits of using diversity weren't used and the space devoted to diversity states was wasted.

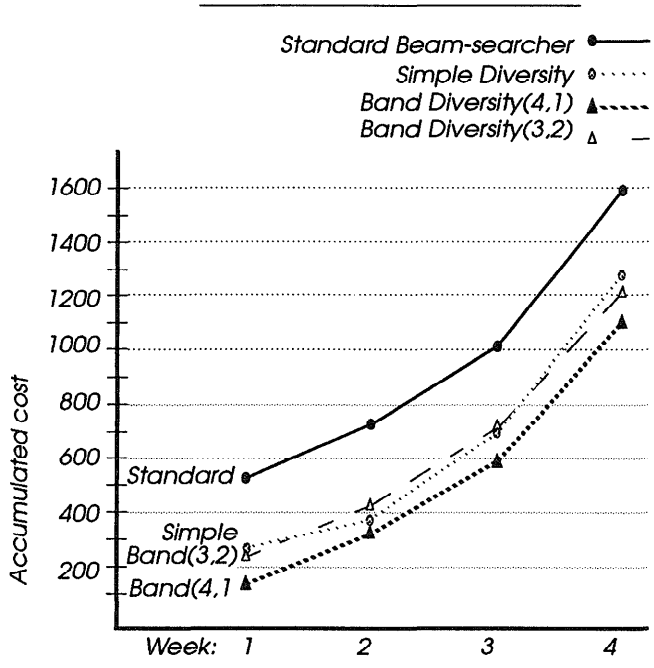

Figure 1: Total cost of the standard beam-search and diversity searchers in the CRESUS *expert system.*

Table 2: Percentage cost of diversity searchers compared to standard beam searcher

Week	1	2	3	4	Average
Standard	*100%*	*100%*	*100%*	*100%*	*100%*
Simple	*52%*	*48%*	*98%*	*112%*	*81%*
Band(4,1)	*28%*	*94%*	*79%*	*102%*	*71%*
Band(3,2)	*50%*	*83%*	*87%*	*98%*	*78%*

Conclusions

Diversity searching is a step towards extending knowledge-based searchers to obtain the benefits of genetic search while retaining the advantages of the symbolic approach. Although the idea of diversity has been investigated in genetic algorithms and simulated annealing, this is the first time that maintaining diversity during a symbolic search has been attempted. This paper addressed the issues involved and presented a framework for diversity search. A domain-independent algorithm was shown and empirical results from a working expert-system show that diversity search can substantially improve knowledge-based searchers.

Diversity search will be most useful in complex domains that are not amenable to genetic algorithms or simulated annealing. These include problems which require a fast response and those which exhibit a high degree of epistasis. It may be that symbolic diversity search will occupy a useful middle-ground between knowledge-rich systems that need to do much search and genetic algorithms and simulated annealing which are not feasible in complex, knowledge-rich domains.

Diversity search shows promise; it warrants more exploration. It should be evaluated with different parameters and combination functions. A simulated-annealing technique of decreasing the diversity as the search progresses according to a schedule may further enhance the search results. It would also be interesting to try applying it to different problems and with different searchers in order to determine how broadly applicable it is.

Acknowledgments

I would like to thank Jaime Carbonell, Michael Mauldin, Ben MacLaren, Rick Chimera and Alex Franz for helpful comments on earlier drafts of this paper, and the rest of the CRESUS team: Javier Berbiela, Maria José Moro Martín, Paloma Bilbao and Lorenzo Tello.

References

Ackley, D.H., Hinton, G.E. and Sejnowski, T.J. 1985. A Learning Algorithm for Boltzmann Machines. *Cognitive Science.* 9(1):147-169.

Beal, Don F. 1990. A Generalized Quiescence Search Algorithm. *Artificial Intelligence.* 43(1):85-98, April.

Booker, L. 1987. Improving Search in Genetic Algorithms. *Genetic Algorithms and Simulated Annealing.* In Davis, L., Morgan Kaufmann Publishers, Los Altos, California, pages 61-73.

Davis, L. and Steenstrup, M. 1987. Genetic Algorithm and Simulated Annealing. *Genetic Algorithms and Simulated Annealing.* In Davis, L., Morgan Kaufmann Publishers, Los Altos, California, pages 1-11.

DeJong, K.A. 1975. *Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ph.D. thesis, University of Michigan, Ann Arbor.

DeJong, K.A. 1990 Special Issue on Genetic Algorithms. *Machine Learning.* 5(4).

Holland, J. 1975. *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Jerrum, M., and Sorkin, G.B. 1993. *Simulated Annealing for Graph Bisection.* Technical Report, University of Edinburgh, Dept. of Computer Science.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. 1983. Optimization by Simulated Annealing. *Science.* (220):671-680.

Mauldin, M.L. 1984. Maintaining Diversity in Genetic Search. *Proceedings of the National Conference on Artificial Intelligence*, pages 247-250.

Newell, Allen. 1978. *Harpy, Production Systems and Human Cognition.* Pittsburgh, PA: Carnegie Mellon University.

Shell, P. and Carbonell, J. G. 1988. *The Parmenides Reference Manual.* CMU Computer Science Department internal paper.

Shell, P. and Carbonell, J. G. 1986. *The FRuleKit Reference Manual.* CMU Computer Science Department internal paper.

Shell, P., et al. 1992. CRESUS: An Integrated Expert System for Cash Management. Scott, A. and Klahr, P. (editor), *Innovative Applications of Artificial Intelligence 4.*