From: SCCC Workshop on Advances & Trends in AI for Problem Solving,
Santiago Chile, November 2000

Hill Climbing for Poker Solitaire: A Case Study on the Impact of Neighborhood Size

Matthew Merzbacher

Mills College, Oakland CA 94613, USA
`matthew@mills.edu`

Abstract. This paper presents the results of an empirical study on the use of broadening and heuristics to improve search. We compare the performance of hill-climbing algorithms with different search breadth. Broadening the alternatives for hill-climbing yields small improvement in search results and is not generally worth the performance penalty incurred. In fact, evidence shows that broadening can degrade results. By contrast, even simple heuristics show great improvement, especially when combined with broadening, leading to conclusions about how to spend limited search resources most effectively.

Keywords: hill-climbing, heuristic search, broadening, gradient ascent

1 Introduction

Gradient ascent, also known as *hill-climbing*, is a fundamental search technique that operates efficiently without needing much memory. Starting with a random position, the best alternative move to a "neighboring" position is taken iteratively until no improvement is possible. With luck, the best possible position, or *optimum*, is found. The problems with gradient ascent are well known, including:

Plateau	which has no gradient, so that all neighbors are equal.
Ridge	where no single step improves things, but where a small step down can be followed by a larger step up.
Local Maximum	where all nearby positions are worse, but which falls short of the optimum.
Steep Optimum	where the best solution is surrounded by relatively unpromising territory, making it unlikely to be encountered.

Several solutions to these problems exist, including *random restart* and *simulated annealing* [2, 3]. Random restart repeatedly runs the hill-climbing algorithm from randomly generated starting positions, with the hope that at least one of the trials will start on the optimal hill. Simulated annealing allows some non-increasing steps to try to escape local maxima, ridges, and plateaus.

Neither random restart nor simulated annealing will usually be lucky enough to solve the steep optimum problem. The best solution in this case is to have some *heuristic* estimate to guide the search. The heuristic estimates the long-term promise of the particular position, rather than its actual score. Thus, the area around the steep optimum, which scores low, can still "look good" heuristically.

Each of these techniques (hill-climbing, simulated annealing, heuristic evaluation) can be augmented by *broadening* the search. Instead of considering just a few neighbors at each iteration, the neighborhood is extended outward to include more candidates. There are several variants of broadening, including macros [4], which consider selected promising more-distant neighbors based on past results. The ultimate broadening considers all possible solutions and will locate the optimum, but with unacceptable consequence to the search space size. Limited broadening may help avoid the problems associated with gradient ascent without overly slowing search.

In order to study the effect of broadening on hill-climbing algorithms, we use a problem with a large search space, many local maxima, and with a steep optimum.

2 The Problem

The particular problem, inspired by a magazine contest [1], uses the game "poker solitaire". In this game, twenty-five cards are dealt from a standard deck in a five-by-five grid. Each row, column and the two diagonals of the grid are evaluated as poker hands, scoring a varying number of points. The goal is to reposition the cards[1], maximizing the sum of the twelve scores (five rows, five columns, two diagonals). The score for each hand is:

pair	(1 point) two of the five cards have the same rank.
two pair	(3 points) two pairs amongst the five cards.
three of a kind	(5 points) three of the five cards have the same rank.
straight	(7 points) the five cards may be arranged in sequence, e.g., 8–6–9–7–5. Aces may be high (above King) or low (below 2).
flush	(10 points) All five cards are the same suit.
full house	(12 points) one pair and three of a kind.
four of a kind	(25 points) four of the five cards have the same rank.
straight-flush	(50 points) both a straight and a flush.

The basic hill-climbing algorithm starts with a random grid of twenty-five cards and then swaps pairs of cards in the grid, as long as the score continues to improve. This algorithm can be broadened allowing consideration of more than two cards at a time for swapping. However, the number of neighbors when k cards are swapped is found by determining the possible combinations of k cards taken from twenty-five and multiplying that by the number of ways those cards can be ordered. For example, when swapping three cards (A, B, and C), A may replace B which replaces C, or A may replace C which replaces B. Removing symmetries, the number of possible moves involving k cards is:

$$B(k) = \frac{25!}{k(25-k)!}$$

---

[1] In the original game of poker solitaire, cards are placed one at a time and may not be moved. The whole point of our game is to allow movement of cards in the grid.

which grows formidably. $B(2) = 300$, $B(3) = 4600$, and and $B(4) = 75900$. Thus, the benefit of broadening will need to be remarkable to outweigh its enormous cost.

This problem also often has a steep optimum. This can be seen by observing that four cards in a straight-flush combined with one bad card scores no points, while swapping that one mismatched card may immediately lead to fifty more points. Even a simple four-flush (four of five cards with the same suit) scores no points, while one swap will score ten. To help solve the steep optimum problem, we employ a simple heuristic estimate with positive values for promising hands that would otherwise score less. Choosing particular values is difficult, since the promise may not pay off, but given the immense relative value of a straight-flush, a four-straight-flush is quite promising and even a three-straight-flush is worth consideration. Experimentally, we demonstrate the value of this simple heuristic in conjunction with broadening.

3 Experiments

To measure the cost and value of broadening and heuristics, 1038 sets of twenty-five cards were randomly selected. For each set, one hundred random starting grids were generated. For each random grid, we ran both regular and heuristic hill-climbing with $k = 2$ and $k = 3$. In the $k = 3$ case, we selected the best choice from all three-way *or* a two-way swaps (thus, the branching factor is really $B(2) + B(3) = 4900$). In the case of heuristic hill-climbing, after the heuristic stops improving we took measurements and continued with the regular hill-climbing algorithm to termination. Thus, there are six variants:

1. hill-climbing with $k = 2$
2. hill-climbing with $k = 3$
3. heuristic hill-climbing with $k = 2$
4. heuristic hill-climbing with $k = 3$
5. heuristic hill-climbing, followed by hill-climbing, both with $k = 2$
6. heuristic hill-climbing, followed by hill-climbing, both with $k = 3$

We measured the distributions for both scores and depths along with statistics about how frequently each version of the algorithm yielded the optimum. Since there is no way to determine the true optimum, we used the best result found on any run by any variant. It is possible that obscure optimums could elude detection.

Fig. 1 compares the final score distributions for each of the six variants (average scores for each method are listed in Fig. 2). Qualitatively, the curves are in three distinct pairs – hill-climbing, heuristic hill-climbing, and heuristic hill-climbing with follow; broadening the search has negligible qualitative effect. The average score increase due to broadening is between four and five points, representing an improvement around five percent.

Further, we discovered that broadening to $k = 3$ only outperforms $k = 2$ in 57% of the hill-climbing runs, with similar results for heuristic variants. In the

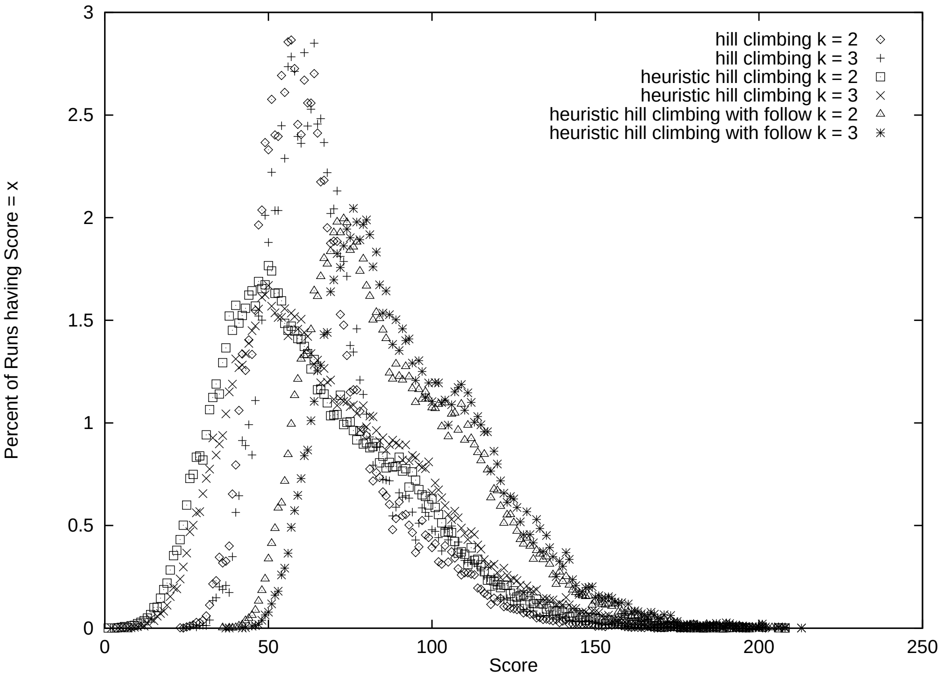

Fig. 1. Score distribution for six versions

other 43%, the narrower hill-climb found as good or better a result. Broadening may not help and was harmful 27% of the time!

Another surprise is that while the heuristic performs poorly on its own, with the followup of hill-climbing it is by far the best. Thus, a very simple heuristic can lead to a spot where blind search can follow up successfully.

method	avg. score	avg. depth	avg. cost	win %
hill-climbing $k = 2$	65.91	9.65	2895	1.15%
hill-climbing $k = 3$	69.36	7.54	36946	6.35%
heur. hill-climbing $k = 2$	64.61	12.61	3783	5.39%
heur. hill-climbing $k = 3$	69.65	10.02	49098	10.11%
heur. hill-climbing w/ follow, $k = 2$	89.01	16.89	5067	34.26%
heur. hill-climbing w/ follow, $k = 3$	93.84	13.39	65611	46.20%

Fig. 2. Algorithm Measurements

The point of the problem, however, was not to find the best average score, but to find the optimum from as many different starting deals as possible. For this task, broadening and heuristics both prove useful, as shown in the final column of Fig. 2. In all variants, broadening leads to more chance of locating the optimum over a run of one hundred tests. Further, this chance rises substantially more when the heuristic was followed by basic hill-climbing.

Let us turn to measuring the depths of the search. The deeper a search goes, the more it costs. Fig. 2 shows the average depth and estimated average cost for each method. The cost is determined by multiplying the depth by the branching factor, since only the single best choice is evaluated at each step. The immense cost of broadening becomes clear here, as the cost with $k = 3$ is approximately thirteen times the cost with $k = 2$. Instead of paying the computational price of broadening, we could use that time for more random restarts. Over a thousand random runs with $k = 2$ can be run for the same price as one hundred runs with $k = 3$.

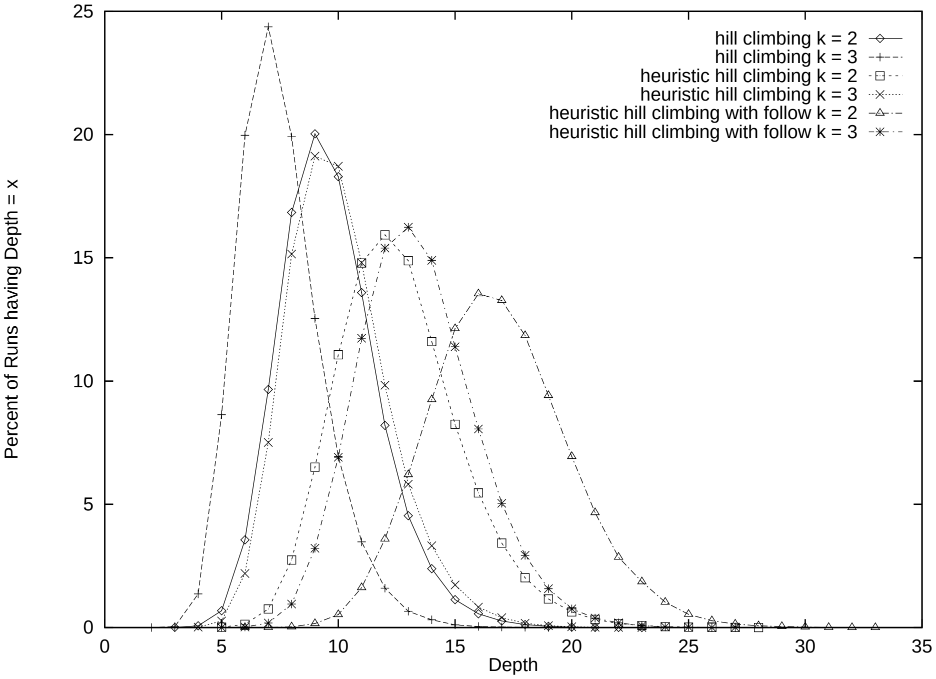

Fig. 3. Distribution of Number of Iterations

Fig. 3 shows the distributions of the number of iterations for each variant. In each scenario, broadening offers approximately twenty percent reduction in the number of iterations, not nearly enough to overcome the additional cost of evaluating so many alternatives.

4 Conclusion

The evidence suggests taking a pessimistic view of broadening. In general, a few well-chosen alternatives [6] will outperform a broad search of all close alternatives. Broadening by a factor of twelve offered about a ten percent overall improvement in the quality of search results. Meanwhile, even a simple heuristic with $k = 2$ dominated searching with $k = 3$.

In practice, the solution to the particular original magazine problem was solved on a Sun SPARCstation 5 taking about a week. With broadening to $k = 3$, the optimal solution was found in a few days, and with heuristics the algorithm took less than an hour. There were several hundred successful entrants to the magazine contest (perhaps calculated by hand, perhaps by computer), so our entry did not win.

Since submission of this paper, we have also used poker solitaire as a homework problem in our introductory artificial intelligence class. The students were required to determine the cost of an optimal solution using breadth-first search and then to identify a promising heuristic. The student heuristics were considerably varied and under comparison in a follow-up experiment.

Improvement due to broadening and improvement due to the use of heuristics seems to be nearly independent. This suggests that broadening could be used in conjunction with heuristics for promising search runs, but not for all cases.

The impact of broadening must also enter into calculations when considering techniques such as macros, which exchange breadth for depth. Broadening can be worthwhile, but must be done with care and discrimination to be useful.

References

1. "Contest: Poker Solitaire II", *Games Magazine*, November 1999.
2. R. Greiner, "PALO: a probabilistic hill-climbing algorithm," *Artificial Intelligence*, vol. 84, no. 1-2, pp 177-208, 1996.
3. S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing," *Science*, 220:671–680, 1983.
4. R. Korf, "Space-Efficient Search Algorithms," *ACM Computing Surveys,* 27(3):337–339, 1996.
5. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Reading MA: Addison-Wesley, 1984.
6. M. Yokoo "Why adding more constraints makes a problem easier for hill-climbing algorithms: analyzing landscapes of CSPs," *Principles and Practice of Constraint Programming – CP '97*, pp. 356–70, Springer–Verlag, 1997.