'Guided' Restarts Hill-Climbing

David Catteeuw, Madalina Drugan, and Bernard Manderick

Artificial Intelligence Lab, Vrije Universiteit Brussel,
Pleinlaan 2, 1050 Brussels, Belgium
{dcatteeu,mdrugan,bmanderi}@vub.ac.be

Abstract. We introduce a new hybrid metaheuristic that combines multiarmed bandits and hill-climbing: 'guided restarts hill-climbing.' We illustrate it on quadratic assignment problem and compare with random restarts hill-climbing.

1 Introduction

Hill-climbing is a local search method for discrete optimization. It starts with a randomly chosen solution, then repeatedly selects the current solution's best neighbor until no improvement is possible—a local optimum is reached. The quality of the resulting solution is highly dependent on the initial one. This is solved by 'random restarts hill-climbing:' repeatedly restart hill-climbing and, when no more time remains, return the best solution found so far.

Drawing initial solutions *uniformly* from the search space is inefficient when some local optima have large basins since many initial solutions lead to the same local optimum. We propose an algorithm which selects initial solutions more intelligently: 'guided restarts hill-climbing.' The algorithm partitions the solution space and learns which regions are worth exploring further and which are not. The partitioning may be problem dependent.

2 Multiarmed Bandits

The problem of selecting a region to explore is similar to a multiarmed bandit—a sequential decision making problem, where an agent must repeatedly select one out of m actions with unknown reward distribution. The agent's goal is to maximize his total reward. He faces the so-called 'exploration-exploitation dilemma.' Should he exploit what he thinks is the best arm and possibly loose out on an even better arm; or explore further hoping to discover a better arm but risking to get poor rewards?

Many learning algorithms address this problem. Some are simple but effective in practice (ϵ-greedy, softmax, and their variants [1]) and often outcompete more complicated algorithms (UCB1, EXP3, and their variants [2, 3]) even though these have theoretical performance guarantees. As an example, we explain ϵ-greedy Q-learning [4]. It estimates the expected reward of each arm with the exponentially weighted moving average of the observed rewards and stores this

in each arm's Q-value. This can be calculated iteratively by incrementing the previous value q_i with $\alpha(r - q_i)$, where α is the learning rate and r is a new reward for arm i. Selecting arms in ϵ-greedy fashion means: selecting the arm with the highest Q-value with high probability $(1 - \epsilon)$, and selecting a random arm with small probability (ϵ, which a parameter of the algorithm). A nice characteristic of Q-learning is that it can quickly track changes in non-stationary reward distributions and its parameters are easy to tune ($\alpha = \epsilon = 0.1$ usually just works).

3 Guided Restarts

'Guided restarts' can be implemented with different multiarmed bandit algorithms. Here, we use ϵ-greedy Q-learning.

1. Parameters: the number of arms m, the learning rate α, and the exploration rate ϵ.
2. The solution space is divided in m regions, each of which corresponds to an arm $i = 1, \ldots, m$. All Q-value are initialized with $Q_0 = 1$.
3. The set of values of the discovered local optima is empty.
4. Repeat for $t = 1, 2, \ldots$ until no more time remains:
   (a) With probability ϵ select an arm at random, with probability $1 - \epsilon$ select the arm with the highest Q-value (ties are broken at random).
   (b) Select a random solution from the corresponding region.
   (c) Perform hill-climbing starting with that solution.
   (d) If the resulting local optimum is the best one seen, store it as the best solution.
   (e) The selected arm i is rewarded with $r = 1 - h/t$, where h is the number of times a local optimum was found with the same value as the current one; the arm's Q-value is updated: $q_i \leftarrow q_i + \alpha(r - q_i)$; and the set of values of discovered local optima is updated.
5. Return the best solution.

4 Constant Improvement to Random Restarts

We applied our algorithm to an instance of the quadratic assignment problem ('Random 20' from [5]): $n = 20$ objects must be assigned to a location, the flow between these objects are given in a matrix F, and the distance between each two locations in a matrix D. A solution is represented by permutation p of n numbers. The goal is find the assignment p^* that minimizes the sum of the products of distance and flow between each two objects: $p^* = \text{argmin}_p \sum_{i=1}^{n} \sum_{j=1}^{n} F_{i,j} D_{p_i,p_j}$, where p_i represents the location of object i.

There are many ways to divide the solution space. We chose an object i at random (at the start of each simulation) and each arm $j = 1, \ldots, n$ assigned object i to a location j: $p_i = j$. We applied the above algorithm with $\alpha = .1$ and $\epsilon = 0$.

As is usually done for problems with solutions represented by permutations, we consider hill-climbing with the neighborhood function '2-exchange:' a solution and its neighbor have the location of two objects swapped.

We performed 100 simulations with both random restarts and guided restarts measuring the number of restarts until a global maximum was found. Guided restarts is (statistically) significantly better than random restarts.[1] The quantile-quantile plot[2] in Fig. 1 shows that guided restarts is a factor 3/2 faster than random restarts.

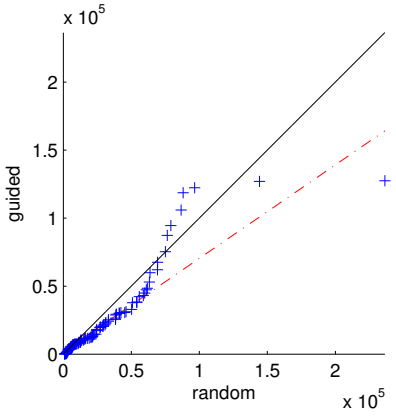

Fig. 1. Quantile-quantile plot of random versus guided restarts. The blue crosses represent the data; the red dashed line represents the trend (connecting first and third quartile); the black solid line is the trend expected from equal algorithms ($y = x$). Guided restarts performs faster than random restarts by a factor 3/2.

5 Conclusion

We introduced guided restarts hill-climbing and successfully demonstrated its advantage over random restarts on a small optimization problem. Guided restarts performed a factor faster than random restarts.

We look forward to applying the method to more challenging problems and to extend it. We think the method can be improved by growing a tree of multiarmed bandits as in Monte Carlo tree search methods [6].

[1] The (Wilcoxon) rank sum test (since the data is not normally but geometrically distributed) rejects the null hypothesis that the median of by random restarts is less than or equal to the median of guided restarts with $p = 0.05$.

[2] In a quantile-quantile plot, the i'th point has coordinates (x_i, y_i) where x_i is the i'th best value of the first sample and y_i the i'th best value of the second sample.

References

1. J. Vermorel and M. Mohri, "Multi-armed Bandit Algorithms and Empirical Evaluation," in *Proceedings of the 16th European Conference on Machine Learning* (J. a. Gama, R. Camacho, P. Brazdil, A. Jorge, and L. Torgo, eds.), Lecture Notes in Computer Science, (Porto, Portugal), pp. 437–448, Springer-Verlag, 2005.
2. P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The Nonstochastic Multiarmed Bandit Problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2003.
3. P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, 2002.
4. C. J. C. H. Watkins, *Learning from Delayed Rewards*. Phd thesis, Cambridge University, 1989.
5. E. Taillard, "Robust taboo search for the quadratic assignment problem," *Parallel Computing*, vol. 17, pp. 443–455, July 1991.
6. C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, pp. 1–49, Mar. 2012.