Stochastic Hill Climbing*

Jason Brownlee
jasonb@CleverAlgorithms.com
The Clever Algorithms Project
http://www.CleverAlgorithms.com

February 2, 2010
Technical Report: CA-TR-20100202-1

Abstract

The Clever Algorithms project aims to describe a large number of Artificial Intelligence algorithms in a complete, consistent, and centralized manner, to improve their general accessibility. The project makes use of a standardized algorithm description template that uses well-defined topics that motivate the collection of specific and useful information about each algorithm described. This report describes the Stochastic Hill Climbing algorithm using the standardized template.

Keywords: `Clever, Algorithms, Description, Optimization, Stochastic, Hill, Climbing`

1 Introduction

The Clever Algorithms project aims to describe a large number of algorithms from the fields of Computational Intelligence, Biologically Inspired Computation, and Metaheuristics in a complete, consistent and centralized manner [2]. The project requires all algorithms to be described using a standardized template that includes a fixed number of sections, each of which is motivated by the presentation of specific information about the technique [3]. This report describes the Stochastic Hill Climbing algorithm using the standardized template. Section 9 highlights some additional and related techniques discovered during the research of this work, and considers the problem of a practitioner requiring guidance to apply a given technique to a problem different to the problem instances presented in the Clever Algorithm project code examples.

2 Name

Stochastic Hill Climbing, SHC, Random Hill Climbing, RHC, Random Mutation Hill Climbing, RMHC

3 Taxonomy

The Stochastic Hill Climbing algorithm is a Stochastic Optimization algorithm and is a Local Optimization algorithm (contrasted to Global Optimization). It is a direct search technique, as it does not require derivatives of the search space. Stochastic Hill Climbing is an extension of deterministic hill climbing algorithms such as Simple Hill Climbing (first-best neighbor), Steepest-Ascent Hill Climbing (best neighbor), and a parent of approaches such as Parallel Hill Climbing and Random-Restart Hill Climbing.

4 Strategy

The strategy of the Stochastic Hill Climbing algorithm is iterate the process of randomly selecting a neighbor for a candidate solution and only accept it if it results in an improvement. The strategy was proposed to address the limitations of deterministic hill climbing techniques that were likely to get stuck in local optima due to their greedy acceptance of neighboring moves.

5 Procedure

Algorithm 1 provides a pseudo-code listing of the Stochastic Hill Climbing algorithm for minimizing a cost function, specifically the Random Mutation Hill Climbing algorithm described by Forrest and Mitchell applied to a maximization optimization problem [6].

Algorithm 1: Pseudo Code Listing for the Stochastic Hill Climbing algorithm.

Input: $Iter_{max}$, ProblemSize
Output: Current

```
Current ← RandomSolution(ProblemSize);
foreach iter_i ∈ Iter_max do
    Candidate ← RandomNeighbor(Current);
    if Cost(Candidate) ≥ Cost(Current) then
        Current ← Candidate;
    end
end
return Current;
```

6 Heuristics

- Stochastic Hill Climbing was designed to be used in discrete domains with explicit neighbors such as combinatorial optimization (compared to continuous function optimization).
- The algorithm's strategy may be applied to continuous domains by making use of a step-size to define candidate-solution neighbors (such as Localized Random Search and Fixed Step-Size Random Search).
- Stochastic Hill Climbing is a local search technique (compared to global search) and may be used to refine a result after the execution of a global search algorithm.
- Even though the technique uses a stochastic process, it can still get stuck in local optima.
- Neighbors with better or equal cost should be accepted, allowing the technique to navigate across plateaus in the response surface.
- The algorithm can be restarted and repeated a number of times after it converges to provide an improved result (called Multiple Restart Hill Climbing).
- The procedure can be applied to multiple candidate solutions concurrently, allowing multiple algorithm runs to be performed at the same time (called Parallel Hill Climbing).

7 Code Listing

Listing 1 provides an example of the Stochastic Hill Climbing algorithm implemented in the Ruby Programming Language, specifically the Random Mutation Hill Climbing algorithm described by Forrest and Mitchell [6]. The algorithm is executed for a fixed number of iterations and is applied to a binary string optimization problem called 'One Max'. The objective of this maximization problem is to prepare a string of all '1' bits, where the cost function only reports the number of bits in a given string.

```
NUM_ITERATIONS = 1000
PROBLEM_SIZE = 64

def cost(bitstring)
  return bitstring.inject(0) {|sum,x| sum = sum + ((x=='1') ? 1 : 0)}
end

def random_solution(problemSize)
  return Array.new(problemSize){|i| (rand<0.5) ? "1" : "0"}
end

def random_neighbor(bitstring)
  mutant = Array.new(bitstring)
  pos = rand(bitstring.length)
  mutant[pos] = (mutant[pos]=='1') ? '0' : '1'
  return mutant
end

def search(numIterations, problemSize)
  candidate = {}
  candidate[:vector] = random_solution(problemSize)
  candidate[:cost] = cost(candidate[:vector])
  numIterations.times do |iter|
    neighbor = {}
    neighbor[:vector] = random_neighbor(candidate[:vector])
    neighbor[:cost] = cost(neighbor[:vector])
    candidate = neighbor if neighbor[:cost] >= candidate[:cost]
    puts " > iteration #{(iter+1)}, best: c=#{candidate[:cost]}, v=#{candidate[:vector].join}"
    break if candidate[:cost] == problemSize
  end
  return candidate
end

best = search(NUM_ITERATIONS, PROBLEM_SIZE)
puts "Done. Best Solution: c=#{best[:cost]}, v=#{best[:vector].join}"
```

Listing 1: Stochastic Hill Climbing algorithm in the Ruby Programming Language

8 References

8.1 Primary Sources

Perhaps the most popular implementation of the Stochastic Hill Climbing algorithm is by Forrest and Mitchell, who proposed the Random Mutation Hill Climbing (RMHC) algorithm (with communication from Richard Palmer) in a study that investigated the behavior of the genetic algorithm on a deceptive class of (discrete) bit-string optimization problem called 'royal road' functions [6]. The RMHC was compared to two other hill climbing algorithms in addition to the genetic algorithm, specifically: the Steepest-Ascent Hill Climber, and the Next-Ascent Hill Climber. This study was then followed up by Mitchell and Holland [9]

Jules and Wattenberg were also early to consider stochastic hill climbing as an approach to compare to the genetic algorithm [8]. Skalak applied the RMHC algorithm to a single long bit-string that represented a number of prototype vectors for use in classification [13].

8.2 Learn More

The Stochastic Hill Climbing algorithm is related to the genetic algorithm without crossover. Simplified version's of the approach are investigated for bit-string based optimization problems with the population size of the genetic algorithm reduced to one. The general technique has been investigated under the names Iterated Hillclimbing [10], ES(1+1,m,hc) [11], Random Bit Climber [5], and (1+1)-Genetic Algorithm [1]. This main difference between RMHC and ES(1+1) is that the latter uses a fixed probability of a mutation for each discrete element of a solution (meaning the neighborhood size is probabilistic), whereas RMHC will only stochastically modify one element.

9 Conclusions

This report described the Stochastic Hill Climbing algorithm as a strategy for addressing some of the limitations of deterministic hill climbing methods. Some additional technique names discovered during the investigation of this work include: *Dynamic Hill Climbing* [15], the investigation of a theoretical abstraction of hill climbing algorithms called *Generalized Hill Climbing* [7, 14], *Probabilistic Hill Climbing* [12] and a second probabilistic hill climbing algorithm called Probably Approximate Locally Optimal (PALO) [4].

A concern highlighted in the research for this report was the guidance needed for a practitioner to apply a technique to a specific problem that differences from the class of problem presented. Guidance may be in the form of a process where articles and papers on, or related to the selected problem are identified, or problem transformations used to map the instance to something that can be addressed by a the chosen technique. This may be included as an advanced topic in the Clever Algorithms project or included in the introductory material.

10 Contribute

Found a typo in the content or a bug in the source code? Are you an expert in this technique and know some facts that could improve the algorithm description for all? Do you want to get that warm feeling from contributing to an open source project? Do you want to see your name as an acknowledgment in print?

Two pillars of this effort are i) that the best domain experts are people outside of the project, and ii) that this work is wrong by default. Please help to make this work less wrong by emailing the author 'Jason Brownlee' at `jasonb@CleverAlgorithms.com` or visit the project website at `http://www.CleverAlgorithms.com`.

References

[1] Thomas Back. Optimal mutation rates in genetic search. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 2–9, 1993.

[2] Jason Brownlee. The clever algorithms project: Overview. Technical Report CA-TR-20100105-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[3] Jason Brownlee. A template for standardized algorithm descriptions. Technical Report CA-TR-20100107-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[4] William W. Cohen, Russell Greiner, and Dale Schuurmans. Probabilistic hill-climbing. In *Computational Learning Theory and Natural Learning Systems*, 1994.

[5] L. Davis. Bit-climbing, representational bias, and test suite design. In *Proceedings of the fourth international conference on genetic algorithms*, pages 18–23, 1991.

[6] Stephanie Forrest and Melanie Mitchell. Relative building-block fitness and the building-block hypothesis. In *Foundations of Genetic Algorithms 2*, pages 109–126. Morgan Kaufmann, 1993.

[7] A. W. Johnson and S. H. Jacobson. On the convergence of generalized hill climbing algorithms. *Discrete Applied Mathematics*, 119(1–2):37–57, 2002.

[8] A. Juels and M. Wattenberg. Stochastic hill climbing as a baseline method for evaluating genetic algorithms. Technical report, University of California, Berkeley, 1994.

[9] Melanie Mitchell and John H. Holland. When will a genetic algorithm outperform hill climbing? In *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., 1993.

[10] Heinz Muhlenbein. Evolution in time and space - the parallel genetic algorithm. In *Foundations of Genetic Algorithms*, 1991.

[11] Heinz Muhlenbein. How genetic algorithms really work: I. mutation and hillclimbing. In *Parallel Problem Solving from Nature 2*, pages 15–26, 1992.

[12] F. Romeo and Alberto L. Sangiovanni-Vincentelli. Probabilistic hill-climbing algorithms: properties and applications. Technical Report UCB/ERL M84/34, EECS Department, University of California, Berkeley, 1984.

[13] David B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the eleventh international conference on machine learning*, pages 293–301. Morgan Kaufmann, 1994.

[14] K. A. Sullivan and S. H. Jacobson. A convergence analysis of generalized hill climbing algorithms. *IEEE Transactions on Automatic Control*, 46(8):1288–1293, August 2001.

[15] Deniz Yuret and Michael de la Maza. Dynamic hill climbing: Overcoming the limitations of optimization techniques. In *The Second Turkish Symposium on Artificial Intelligence and Neural Networks*, 1993.