Simulated Annealing*

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

November 15, 2010
Technical Report: CA-TR-20101115c-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 Simulated Annealing algorithm.

Keywords: `Clever, Algorithms, Description, Optimization, Simulated, Annealing`

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 Simulated Annealing algorithm.

2 Name

Simulated Annealing, SA

3 Taxonomy

Simulated Annealing is a global optimization that belongs to the field of Stochastic Optimization and Metaheuristics. Simulated Annealing is an adaptation of the Metropolis-Hastings Monte Carlo algorithm and is used in function optimization. Like the Genetic Algorithm, it provides a basis for a large variety of extensions and specialization's of the general method not limited to Parallel Simulated Annealing, Fast Simulated Annealing, and Adaptive Simulated Annealing.

4 Inspiration

Simulated Annealing is inspired by the process of annealing in metallurgy. In this natural process a material is heated and slowly cooled under controlled conditions to increase the size of the crystals in the material and reduce their defects. This has the effect of improving the strength and durability of the material. The heat increases the energy of the atoms allowing them to move freely, and the slow cooling schedule allows a new low-energy configuration to be discovered and exploited.

5 Metaphor

Each configuration of a solution in the search space represents a different internal energy of the system. Heating the system results in a relaxation of the acceptance criteria of the samples taken from the search space. As the system is cooled, the acceptance criteria of samples is narrowed to focus on improving movements. Once the system has cooled, the configuration will represent a sample at or close to a global optimum.

6 Strategy

The information processing objective of the technique is to locate the minimum cost configuration in the search space. The algorithms plan of action is to probabilistically re-sample the problem space where the acceptance of new samples into the currently held sample is managed by a probabilistic function that becomes more discerning of the cost of samples it accepts over the execution time of the algorithm. This probabilistic decision is based on the Metropolis-Hastings algorithm for simulating samples from a thermodynamic system.

7 Procedure

Algorithm 1 provides a pseudo-code listing of the main Simulated Annealing algorithm for minimizing a cost function.

Algorithm 1: Pseudo Code for the Simulated Annealing algorithm.

```
Input: ProblemSize, $iterations_{max}$, $temp_{max}$
Output: $S_{best}$
$S_{current} \leftarrow$ CreateInitialSolution(ProblemSize);
$S_{best} \leftarrow S_{current}$;
for $i = 1$ to $iterations_{max}$ do
    $S_i \leftarrow$ CreateNeighborSolution($S_{current}$);
    $temp_{curr} \leftarrow$ CalculateTemperature($i$, $temp_{max}$);
    if Cost($S_i$) $\leq$ Cost($S_{current}$) then
        $S_{current} \leftarrow S_i$;
        if Cost($S_i$) $\leq$ Cost($S_{best}$) then
            $S_{best} \leftarrow S_i$;
        end
    else if $\exp(\frac{\text{Cost}(S_{current}) - \text{Cost}(S_i)}{temp_{curr}}) >$ Rand() then
        $S_{current} \leftarrow S_i$;
    end
end
return $S_{best}$;
```

8 Heuristics

- Simulated Annealing was designed for use with combinatorial optimization problems, although it has been adapted for continuous function optimization problems.
- The convergence proof suggests that with a long enough cooling period, the system will always converge to the global optimum. The downside of this theoretical finding is that the number of samples taken for optimum convergence to occur on some problems may be more than a complete enumeration of the search space.
- Performance improvements can be given with the selection of a candidate move generation scheme (neighborhood) that is less likely to generate candidates of significantly higher cost.
- Restarting the cooling scheduling using the best found solution so far can lead to an improved outcome on some problems.
- A common acceptance method is to always accept improving solutions and accept worse solutions with a probability of $P(accept) \leftarrow \exp(\frac{e-e\prime}{T})$, where T is the current temperature, e is the energy (or cost) of the current solution and $e\prime$ is the energy of a candidate solution being considered.
- The size of the neighborhood considered in generating candidate solutions may also change over time or be influenced by the temperature, starting initially broad and narrowing with the execution of the algorithm.
- A problem specific heuristic method can be used to provide the starting point for the search.

9 Code Listing

Listing 1 provides an example of the Simulated Annealing algorithm implemented in the Ruby Programming Language. The algorithm is applied to the Berlin52 instance of the Traveling Salesman Problem (TSP), taken from the TSPLIB. The problem seeks a permutation of the order to visit cities (called a tour) that minimized the total distance traveled. The optimal tour distance for Berlin52 instance is 7542 units.

The algorithm implementation uses a two-opt procedure for the neighborhood function and the classical $P(accept) \leftarrow \exp(\frac{e-e\prime}{T})$ as the acceptance function. A simple linear cooling regime is used with a large initial temperature which is decreased each iteration. The initial solution is created using a nearest neighbor heuristic to provide a good starting point for the search.

```ruby
def euc_2d(c1, c2)
  Math.sqrt((c1[0] - c2[0])**2.0 + (c1[1] - c2[1])**2.0).round
end

def cost(permutation, cities)
  distance =0
  permutation.each_with_index do |c1, i|
    c2 = (i==permutation.length-1) ? permutation[0] : permutation[i+1]
    distance += euc_2d(cities[c1], cities[c2])
  end
  return distance
end

def nearest_neighbor_solution(cities)
  candidate = {}
  candidate[:vector] = [rand(cities.length)]
  all_cities = Array.new(cities.length) {|i| i}
  while candidate[:vector].length < cities.length
    next_city = {:city=>nil,:dist=>nil}
    candidates = all_cities - candidate[:vector]
    candidates.each do |city|
      dist = euc_2d(cities[candidate[:vector].last], city)
      if next_city[:city].nil? or next_city[:dist] < dist
        next_city[:city] = city
        next_city[:dist] = dist
      end
    end
    candidate[:vector] << next_city[:city]
  end
  candidate[:cost] = cost(candidate[:vector], cities)
  return candidate
end

def two_opt!(perm)
  c1, c2 = rand(perm.length), rand(perm.length)
  c2 = rand(perm.length) while c1 == c2
  c1, c2 = c2, c1 if c2 < c1
  perm[c1...c2] = perm[c1...c2].reverse
  return perm
end

def create_neighbour(current, cities)
  candidate = {}
  candidate[:vector] = Array.new(current[:vector])
  two_opt!(candidate[:vector])
  two_opt!(candidate[:vector])
  candidate[:cost] = cost(candidate[:vector], cities)
  return candidate
end

def should_accept?(candidate, current, temp)
  return true if candidate[:cost] <= current[:cost]
  return Math.exp((current[:cost] - candidate[:cost]) / temp) > rand()
end

def search(cities, max_iter, max_temp, temp_change)
  current, temp = nearest_neighbor_solution(cities), max_temp
  best = current
  puts "Nearest Neighbor heuristic solution: cost=#{current[:cost]}"
  max_iter.times do |iter|
    candidate = create_neighbour(current, cities)
    temp = temp * temp_change
    current = candidate if should_accept?(candidate, current, temp)
    best = candidate if candidate[:cost] < best[:cost]
    puts " > iteration #{(iter+1)}, temp=#{temp}, best=#{best[:cost]}"
  end
  return best
end

if __FILE__ == $0
  berlin52 = [[565,575],[25,185],[345,750],[945,685],[845,655],[880,660],[25,230],
   [525,1000],[580,1175],[650,1130],[1605,620],[1220,580],[1465,200],[1530,5],
   [845,680],[725,370],[145,665],[415,635],[510,875],[560,365],[300,465],
   [520,585],[480,415],[835,625],[975,580],[1215,245],[1320,315],[1250,400],
   [660,180],[410,250],[420,555],[575,665],[1150,1160],[700,580],[685,595],
   [685,610],[770,610],[795,645],[720,635],[760,650],[475,960],[95,260],
   [875,920],[700,500],[555,815],[830,485],[1170,65],[830,610],[605,625],
   [595,360],[1340,725],[1740,245]]
  max_iterations = 5000
  max_temp = 100000.0
  temp_change = 0.98

  best = search(berlin52, max_iterations, max_temp, temp_change)
  puts "Done. Best Solution: c=#{best[:cost]}, v=#{best[:vector].inspect}"
end
```

Listing 1: Simulated Annealing algorithm in the Ruby Programming Language

10 References

10.1 Primary Sources

Simulated Annealing is credited to Kirkpatrick, Gelatt, and Vecchi in 1983 [6]. Granville, Krivanek, and Rasson provided the proof for convergence for Simulated Annealing in 1994 [4]. There were a number of early studies and application papers such as Kirkpatrick's investigation into the TSP and minimum cut problems [7], and a study by Vecchi and Kirkpatrick on Simulated Annealing applied to the global wiring problem [9].

10.2 Learn More

There are many excellent reviews of Simulated Annealing, not limited to the review by Ingber that describes improved methods such as Adaptive Simulated Annealing, Simulated Quenching, and hybrid methods [5]. There are books dedicated to Simulated Annealing, applications and variations. Two examples of good texts include "Simulated Annealing: Theory and Applications" by Laarhoven and Aarts [8] that provides an introduction to the technique and applications, and "Simulated Annealing: Parallelization Techniques" by Robert Azencott [1] that focuses of the theory and applications of parallel methods for Simulated Annealing.

11 Conclusions

This report described the Simulated Annealing algorithm using the standardized template.

12 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 subjected to continuous improvement. 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] Robert Azencott. *Simulated annealing: parallelization techniques.* Wiley, 1992.

[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] J.-P. Rasson Granville, V.; M. Krivanek. Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, 1994.

[5] Lester Ingber. Simulated annealing: Practice versus theory. *Math. Comput. Modelling*, 18, 1993.

[6] M. P. Vecchi Kirkpatrick, S.; C. D. Gelatt. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[7] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *JOURNAL OF STATISTICAL PHYSICS*, 34:975–986, 1983.

[8] Peter J. M. van Laarhoven and Emile H. L. Aarts. *Simulated Annealing: Theory and Applications*. Springer, 1988.

[9] M.P. Vecchi and S. Kirkpatrick. Global wiring by simulated annealing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2(4):215–222, 1983.