Bees Algorithm*

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

November 20, 2010
Technical Report: CA-TR-20101120a-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 Bees Algorithm using the standardized algorithm template.

Keywords: `Clever, Algorithms, Description, Optimization, Bees, Algorithm`

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 [1]. 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 [2]. This report describes the Bees Algorithm using the standardized algorithm template.

2 Name

Bees Algorithm, BA

3 Taxonomy

The Bees Algorithm beings to Bee Inspired Algorithms and the field of Swarm Intelligence, and more broadly the fields of Computational Intelligence and Metaheuristics. The Bees Algorithm is related to other Bee Inspired Algorithms, such as Bee Colony Optimization, and other Swarm Intelligence algorithms such as Ant Colony Optimization and Particle Swarm Optimization.

4 Inspiration

The Bees Algorithm is inspired by the foraging behavior of honey bees. Honey bees collect nectar from vast areas around their hive (more than 10 kilometers). The have been observed to send bees to collect nectar from patches relative to the amount of food available at each flower patch. Finally, bees communicate with each other via a waggle dance that informs other bees in the hive as to the direction, distance, and quality rating of food sources.

5 Metaphor

Honey bees collect nectar from flower patches as a food source for the hive. The hive sends out scout's that locate patches of flowers, then return to the hive and inform other bees as to the fitness and location of a food source via a waggle dance. The scout returns to the flower patch with follower bees. A small number of scouts continue to search for new patches, while bees returning from flower patches continue to communicate the quality of the patch.

6 Strategy

The information processing objective of the algorithm is to locate and explore good sites within a problem search space. Scouts are sent out to randomly sample the problem space and locate good sites. The good sites are exploited via the application of a local search, where a smaller number of good sites are explored more than the others. Good sites are continually explored, although many scouts are sent out each iteration always in search of additional good sites.

7 Procedure

Algorithm 1 provides a pseudo-code listing of the Bees Algorithm for minimizing a cost function.

8 Heuristics

- The Bees Algorithm was developed to be used with continuous and combinatorial function optimization problems.
- The $Patch_{size}$ variable is used as the neighborhood size. For example, in a continuous function optimization problem, each dimension of a site would be sampled as $x_i \pm (rand() \times Patch_{size})$.
- The $Patch_{size}$ variable is decrease each iteration, typically by a constant amount (such as 0.95).
- The number of elite sites ($EliteSites_{num}$) must be $<$ the number of sites ($Sites_{num}$), and the number of elite bees ($EliteBees_{num}$) is traditionally $<$ the number of other bees ($OtherBees_{num}$).

9 Code Listing

Listing 1 provides an example of the Bees Algorithm implemented in the Ruby Programming Language. The demonstration problem is an instance of a continuous function optimization that seeks $minf(x)$ where $f = \sum_{i=1}^{n} x_i^2$, $-5.0 \leq x_i \leq 5.0$ and $n = 3$. The optimal solution for this basin function is $(v_0, \ldots, v_{n-1}) = 0.0$. The algorithm is an implementation of the Bees Algorithm as described in the seminal paper [4]. A fixed patch size decrease factor of 0.95 was applied each iteration.

Algorithm 1: Pseudo Code for the Bees Algorithm.

Input: $Problem_{size}$, $Bees_{num}$, $Sites_{num}$, $EliteSites_{num}$, $PatchSize_{init}$, $EliteBees_{num}$, $OtherBees_{num}$

Output: Bee_{best}

```
Population ← InitializePopulation(Bees_num, Problem_size);
while ¬StopCondition() do
    EvaluatePopulation(Population);
    Bee_best ← GetBestSolution(Population);
    NextGeneration ← 0;
    Patch_size ← ( PatchSize_init × PatchDecrease_factor);
    Sites_best ← SelectBestSites(Population, Sites_num);
    foreach Site_i ∈ Sites_best do
        RecruitedBees_num ← 0;
        if i < EliteSites_num then
            RecruitedBees_num ← EliteBees_num;
        else
            RecruitedBees_num ← OtherBees_num;
        end
        Neighborhood ← 0;
        for j to RecruitedBees_num do
            Neighborhood ← CreateNeighbourhoodBee(Site_i, Patch_size);
        end
        NextGeneration ← GetBestSolution(Neighborhood);
    end
    RemainingBees_num ← (Bees_num- Sites_num);
    for j to RemainingBees_num do
        NextGeneration ← CreateRandomBee();
    end
    Population ← NextGeneration;
end
return Bee_best;
```

```ruby
def objective_function(vector)
  return vector.inject(0.0) {|sum, x| sum + (x ** 2.0)}
end

def random_vector(problem_size, search_space)
  return Array.new(problem_size) do |i|
    search_space[i][0] + ((search_space[i][1] - search_space[i][0]) * rand())
  end
end

def create_random_bee(problem_size, search_space)
  bee = {}
  bee[:vector] = random_vector(problem_size, search_space)
  return bee
end

def create_neighborhood_bee(site, patch_size, search_space)
  vector = []
  site.each_with_index do |v,i|
    v = (rand()<0.5) ? v+rand()*patch_size : v-rand()*patch_size
    v = search_space[i][0] if v < search_space[i][0]
```

```ruby
    v = search_space[i][1] if v > search_space[i][1]
    vector << v
  end
  bee = {}
  bee[:vector] = vector
  return bee
end

def search_neighborhood(site, neighborhood_size, patch_size, search_space)
  neighborhood = []
  neighborhood_size.times do
    neighborhood << create_neighborhood_bee(site[:vector], patch_size, search_space)
  end
  neighborhood.each{|bee| bee[:fitness] = objective_function(bee[:vector])}
  return neighborhood.sort{|x,y| x[:fitness]<=>y[:fitness]}.first
end

def search(max_gens, problem_size, search_space, num_bees, num_sites, elite_sites, patch_size,
    e_bees, o_bees)
  best = nil
  pop = Array.new(num_bees){ create_random_bee(problem_size, search_space) }
  max_gens.times do |gen|
    pop.each{|bee| bee[:fitness] = objective_function(bee[:vector])}
    pop.sort!{|x,y| x[:fitness]<=>y[:fitness]}
    best = pop.first if best.nil? or pop.first[:fitness] < best[:fitness]
    next_generation = []
    pop[0...num_sites].each_with_index do |site, i|
      neighborhood_size = (i<elite_sites) ? e_bees : o_bees
      next_generation << search_neighborhood(site, neighborhood_size, patch_size, search_space)
    end
    (num_bees-num_sites).times do
      next_generation << create_random_bee(problem_size, search_space)
    end
    pop = next_generation
    patch_size = patch_size * 0.95
    puts " > iteration=#{gen+1}, patch_size=#{patch_size}, fitness=#{best[:fitness]}"
  end
  return best
end

if __FILE__ == $0
  problem_size = 3
  search_space = Array.new(problem_size) {|i| [-5, 5]}
  max_gens = 500
  num_bees = 45
  num_sites = 3
  elite_sites = 1
  patch_size = 3.0
  e_bees = 7
  o_bees = 2

  best = search(max_gens, problem_size, search_space, num_bees, num_sites, elite_sites,
      patch_size, e_bees, o_bees)
  puts "done! Solution: f=#{best[:fitness]}, s=#{best[:vector].inspect}"
end
```

Listing 1: Bees Algorithm in the Ruby Programming Language

10 References

10.1 Primary Sources

The Bees Algorithm was proposed by Pham, et al. in a technical report in 2005 [5], and later published [4]. In this work, the algorithm was applied to standard instances of continuous function optimization problems.

10.2 Learn More

The majority of the work on the algorithm has concerned its application to various problem domains. The following is a selection of popular application papers: the optimization of linear antenna arrays by Guney and Onay [3], the optimization of codebook vectors in the Learning Vector Quantization algorithm for classification by Pham, et al. [7], optimization of neural networks for classification by Pham, et al. [8], and the optimization of clustering methods by Pham, et al. [6].

11 Conclusions

This report described the Bees Algorithm using the standardized algorithm 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] Jason Brownlee. The clever algorithms project: Overview. Technical Report CA-TR-20100105-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

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

[3] K. Guney and M. Onay. Amplitude-only pattern nulling of linear antenna arrays with the use of bees algorithm. *Progress In Electromagnetics Research*, 70:21–36, 2007.

[4] D. T. Pham, Ghanbarzadeh A., Koc E., Otri S., Rahim S., and M.Zaidi. The bees algorithm - a novel tool for complex optimisation problems. In *Proceedings of IPROMS 2006 Conference*, pages 454–461, 2006.

[5] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm. Technical report, Manufacturing Engineering Centre, Cardiff University, 2005.

[6] D. T. Pham, S. Otri, A. A. Afify, M. Mahmuddin, and H. Al-Jabbouli. Data clustering using the bees algorithm. In *Proc 40th CIRP International Manufacturing Systems Seminar*, 2007.

[7] D. T. Pham, S. Otri, A. Ghanbarzadeh, and E. Koc. Application of the bees algorithm to the training of learning vector quantisation networks for control chart pattern recognition. In *Proceedings of Information and Communication Technologies (ICTTA'06)*, pages 1624–1629, 2006.

[8] D. T. Pham, A. J. Soroka, A. Ghanbarzadeh, E. Koc, S. Otri, and M. Packianather. Optimising neural networks for identification of wood defects using the bees algorithm. In *Proceedings of the 2006 IEEE International Conference on Industrial Informatics*, 2006.