10.1109/ICICISYS.2009.5357838

Human-Inspired Algorithms for Continuous Function Optimization

Luna Mingyi Zhang, Cheyenne Dahlmann
The Center for Advanced Studies in Science, Math and Technology of Joseph Wheeler High School
375 Holt Road
Marietta, Georgia 30068, USA
mingyiluna@yahoo.com, cheydahlmann@aol.com

Yanqing Zhang
Department of Computer Science
Georgia State University
P.O. Box 3994
Atlanta, GA 30302-3994, USA
yzhang@cs.gsu.edu

***Abstract*—The Human-Inspired Algorithm (HIA) is a new algorithm that uses a given population (a group of candidate solutions) to improve the search for optimal solutions to continuous functions in different optimization applications such as non-linear programming. HIA imitates the intelligent search strategies of mountain climbers who use modern techniques (such as binoculars and cell phones) to effectively find the highest mountain in a given region. Different from Genetic Algorithms (GAs) and Bees Algorithms (BAs), HIA divides a whole search space into multiple equal subspaces, evenly assigns the population in the subspaces, finds an elite subspace with the largest sum of function values, and uses more climbers (candidate solutions) to explore the elite subspace and fewer ones to explore the rest of the whole search space. BAs use random search in local neighborhood search, whereas HIA uses GAs in local neighborhood search to obtain better results. HIA locates a point with the largest function value among the elite sites and creates a hypercube with the point as its center. The assigned climbers in the hypercube and the elite subspace continue to search for the optimal solution iteratively. In each loop, the hypercube and the elite subspace become smaller to have a larger chance to pinpoint the optimal solution. Simulation results for three continuous functions with constraints and three continuous functions with box constraints can indicate that HIA is more efficient than GAs and BAs. Finally, conclusions and future works are given.**

Keywords- Human-Inspired Algorithms; Swarm Intelligence; Bees Algorithms; Genetic Algorithms; Optimization

I. INTRODUCTION

Swarm intelligence techniques [1][2], such as particle swarm optimization [1][2], bees algorithms [3][4][5], and genetic algorithms [6][7], are useful methods to optimize complex and continuous functions. In general, human beings have a higher level of intelligence than other animals such as bees and ants. Incorporating the intelligence of people into optimization methods has not been studied as much as using the intelligence of other animals in them. Therefore, this paper presents a new optimization algorithm that uses human problem-solving strategies to effectively find optimal or near-optimal solutions to continuous functions.

Particle Swarm Optimization (PSO) is an important population-based stochastic optimization method; it is inspired by the nature of swarms or groups of animals, such as the schooling of fish and the flocking of birds [1][2]. Possible solutions in optimizing functions are called "particles." In small groups, they follow currently the best solutions. Acceleration, velocity, and random values are a few factors that determine how the particles move toward their best locations. Although both PSO and GA use evolutionary computations, PSO has no operators, such as mutation and crossover [1][2].

The bees algorithm (BA) imitates the behavior of honey bees when the bees try to locate the best food source [3][4][5]. The algorithm uses neighborhood and random searches to find optimal or near-optimal solutions to continuous functions. When a bee randomly finds a source of food or nectar, the bee performs a dance called the "waggle dance" to inform the other bees of the properties of the source of nectar, such as quality and abundance. The location of the source is called a site. Elite sites are few best sites where more bees are recruited to conduct random search. The bees decide on the number of elite sites and on the number of bees to be randomly assigned to each of the elite sites, depending on its quality and distance from the bees' home [3][4][5]. The Bees Algorithm [3][4][5] is briefly shown below.

1. Randomly initialize population.
2. Evaluate function values for the population.
3. While (stop criterion is not met)
   Select elite sites for neighborhood search.
   Recruit bees for the elite sites and evaluate the function value for each bee.
   Select the bee with the largest function value from each site.
   Assign remaining bees to conduct random search.
   Evaluate their function values.
4. End While.

Similar to particle swarm optimization techniques, genetic algorithms (GAs) are "stochastic iterative algorithms" [6]. These algorithms are inspired by the concept of evolution and genetics [6][7]. GAs use heuristics to conduct search; an initial group of individuals or called "chromosomes" is created randomly. Then all individuals are evaluated according to the

978-1-4244-4738-1/09/$25.00 ©2009 IEEE

fitness function. Some important operators are mutation and crossover; their purpose is to create a new population (offsprings) that is better than the old population, so the new solutions are chosen based on their fitness (level of ability to reproduce). [6][7].

To improve particle swarm optimization, bees algorithms and genetic algorithms, a new algorithm, called Human-Inspired Algorithm (HIA), is proposed to find better solutions to complex and continuous optimization problems. HIA is based on the abilities of mountain climbers to quickly find the tallest mountain. In addition, communication among people is much faster than communication among bees, and people are able to analyze and think about how to develop a good plan of distributing everyone instead of just randomly climb anywhere. In general, it is more effective to incorporate some good ideas from particle swarm optimization methods such as BAs and GAs into HIA. Because humans are very intelligent and have technological devices and other materials, they can efficiently find the optimal or near-optimal solution to a function, either constrained or non-constrained.

II. HUMAN-INSPIRED ALGORITHM

Let $f(x_i)$ be a function where $a_i \le x_i \le b_i$ for i=1, 2, …, n. The goal is to maximize $f(x_i)$ in the whole search space U ($a_i \le x_i \le b_i$ for i=1, 2, …, n). This search problem is to find the optimal point ($x_1^*, x_2^*, \ldots, x_n^*$) such that $f(x_i^*)$ for i=1, 2, …, n is the maximum value in U. Let m be the number of equal partitions on dimension X_i. Let N be the population size (the number of climbers that are points in a given space). Let P_{ij} be the sum of function values on x_i for each partition j for i=1, 2, …, n and j=1, 2, …, m. Let S be a subspace of U where it is highly possible to find the optimal solution. Let *max_point* be the current optimal point ($x_1^+, x_2^+, \ldots, x_n^+$).

The human-inspired initialization algorithm is shown in Figure 1. Its purpose is to create a hypercube with *max_point* as its center and S. The hypercube, called H, is a subspace of U where it is highly possible to find the optimal solution. Both H and S will be used by the human-inspired algorithm as shown in Figure 2.

Let L be the number of loops. Let counter_narrow be a counter that keeps track of the number of times that entire search space becomes smaller and let num_loops-1 be the maximum number of loops that the entire search space becomes smaller.

The human-inspired initialization method and the human-inspired method are given below.

Procedure: Human-inspired Initialization(U)
Begin
Initially, space S = the whole search space U.
For i=1 to i=n
Along dimension X_i, divide S into m equal subspaces and evenly distribute the population in the m equal subspaces. Calculate P_{ij} for j=1, 2, …, m. Find a temporary subspace T with max(P_{ij}), where $a_i^1 \le x_i \le b_i^1$. S=T.
End For Loop
Run GA in S for K times to find K candidate best points ($x_1^k, x_2^k, \ldots, x_n^k$) for k=1, 2, …, K such that $f(x_i^k)$ for i=1, 2, …, n is the maximum value under a number of generations. Update *max_point*.
For l=1 to l=L
Run GA to find a candidate best point in S and in U, respectively. Create K+2 hypercubes with K+2 candidate best points as their centers. Assign climbers (points) to the K+2 hypercubes. Run GA in K+2 hypercubes and find K candidate best points. Update *max_point*.
End For Loop
Run GA in U with the new population including K+2 candidate best points and *max_point*; update *max_point*.
Create a hypercube, H, with *max_point* as its center.
End

Procedure: human-inspired algorithm( )
Begin
Human-inspired Initialization(U) shown in Figure 1.
While counter_narrow<number_global_loops
Step 1: Use the hypercube H to find *max_point*.
For i=1 to i=n
Along X_i, divide H into m equal subspaces and evenly distribute the population in them. Calculate P_{ij} for j=1, 2, …, m. Find a subspace T with max(P_{ij}), where $a_i^1 \le x_i \le b_i^1$. H=T.
End For Loop
Run GA in H to find K candidate best points. Update *max_point*.
For l=1 to l=L
Run GA to find a candidate best point in H and in U, respectively. Create K+2 hypercubes; assign climbers (points) to them. Run GA in K+2 hypercubes; find K candidate best points. Update *max_point*.
End For Loop
Run GA in U with the new population including K+2 candidate best points and *max_point*; update *max_point*.

Step 2: Use the current S to find *max_point* and a smaller S.
For i=1 to i=n
Along X_i, divide S into m equal subspaces and evenly distribute the population in them. Calculate P_{ij} for j=1, 2, …, m. Find a subspace T with max(P_{ij}), where $a_i^1 \le x_i \le b_i^1$. S=T.
End For Loop
Run GA in S to find K candidate best points. Update *max_point*.
For l=1 to l=L
Run GA to find a candidate best point in S and in U, respectively. Create K+2 hypercubes; assign climbers (points) to them. Run GA in K+2 hypercubes; find K candidate best points. Update *max_point*.
End For Loop

Run GA in U with the new population including K+2 candidate best points and *max_point*; find *max_point*. Create a hypercube H with *max_point* as its center.

End While Loop

End

III. SIMULATIONS

A. Maximization of Continuous Functions with Constraints

Runarsson and Yao have used evolutionary computing methods to solve 13 commonly used benchmark optimization problems (g01 – g13) [8][9]. Very recently, Tessem and Yen have used the adaptive penalty formulation method for solving g01 – g13 [10]. We choose three different problems (g03, g06 and g08) for simulations.

1. g03

function: $f(x) = \left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i$

constraints: $g(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$ where n=10 and $0 \le x_i \le 1$ (i=1, …, n).

2. g06

function: $f(x) = -(x_1 - 10)^3 - (x_2 - 20)^3$

constraints: $g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$

$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$

$13 \le x_1 \le 100$ and $0 \le x_2 \le 100$.

3. g08

function: $f(x) = \dfrac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$

constraints: $g_1(x) = x_1^2 - x_2 + 1 \le 0$

$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0$

$0 \le x_1 \le 10$ and $0 \le x_2 \le 10$.

Table 1. Comparisons of Results (Optimal Solutions)

Function	Optimal	HIA	BA	GA
g03	1.0000	1.0000	0.99860	0.75729
g06	6961.814	6961.814	6944.733	6934.976
g08	0.095825	0.095825	0.095824	0.095398

Table 2. Comparisons of Results (Average Solutions)

Function	Optimal	HIA	BA	GA
g03	1.0000	1.0000	0.99810	0.58221
g06	6961.814	6961.805	6934.987	6906.429
g08	0.095825	0.095825	0.095824	0.091371

All of the values in Table 1 for HIA, BA, and GA are the largest ones found with the same number of simulations. For each of the three functions, the values obtained for HIA in many simulations do not vary a lot; all are very close to the optimal solutions. However, for BA and GA, their values vary more, especially those in GA. For all 3 functions, HIA found the optimal solutions. The values under HIA, BA, and GA in Table 2 are average function values for 5 simulations. Again, for all three functions, two average values for HIA are the optimal solutions. Thus, HIA has better performance than both BA and GA.

B. Maximization of Continuous Functions with Box Constraints

The proposed algorithm is tested on 3 problems as shown below.

1. f01 (g03 with box constraints)

2. f02

function: $f(x) = x_1 - x_2^2 + x_3^3 - x_4^4 + x_5^5 - x_6^6 + x_7^7 - x_8^8 + x_9^9 - x_{10}^{10}$

where $0 \le x_i \le 2$ (i=1, …, 10).

3. f03

function:

$f(x) = x_1 - x_2^2 + \sin(x_3^3) - \cos(x_4^4) + \sin(x_5^5) - \cos(x_6^6) + \sin(x_7^7)$

$-\cos(x_8^8) + \sin(x_9^9) - \cos(x_{10}^{10})$

where $0 \le x_i \le 2$ (i=1, …, 10).

Table 3. Comparisons of Results (Optimal Solutions)

Function	Optimal	HIA	BA	GA
f01	100000	99997	95771	9348
f02	unknown	681.99	679.65	629.68
f03	unknown	9.9993	9.8715	5.1500

Table 4. Comparisons of Results (Average Solutions)

Function	Optimal	HIA	BA	GA
f01	100000	99993	94603	5618
f02	unknown	681.95	669.45	610.92
f03	unknown	9.9990	9.7890	3.7360

The optimal function values for f02 and f03 are unknown. Similar to Table 1, all of the values in Table 3 for HIA, BA, and GA are the largest ones found in the same number of simulations. For each of the three functions, the values obtained for HIA in many simulations do not vary a lot; all are very close to the optimal solutions. However, for BA and GA, their values vary more, especially those in GA. f01 is changed a little so that there are only box constraints in g03. For f01, the average value for HIA is very close to the optimal solution. The values under HIA, BA, and GA in Table 4 are average function values for 5 simulations. For all three functions, the average values for HIA are larger than those for both BA and GA. Thus, HIA has better performance than both BA and GA.

C. Maximization of g06 under Different Conditions

To evaluate HIA, GA and BA sufficiently and objectively, g06 is used to test these different algorithms under different

conditions (population sizes, numbers of loops, and execution times). Simulation results are given in Tables 5, 6 and 7. The execution time of BA or GA is slightly larger than that of HIA. We can conclude that HIA is more effective and more efficient than GA and BA.

Table 5. Comparisons (Average Solutions) Based on Population Sizes for g06

Population	Optimal	HIA	BA	GA
250	6961.814	6961.787	6920.790	6847.731
500	6961.814	6961.800	6913.881	6896.870
750	6961.814	6961.810	6917.524	6862.283

Table 6. Comparisons (Average Solutions) Based on Number of Loops for g06

No. of Loops	Optimal	HIA	BA	GA
81	6961.814	6961.764	6932.830	6886.246
151	6961.814	6961.789	6932.059	6855.291
221	6961.814	6961.793	6916.774	6882.724

Table 7. Comparisons Based on Execution Time for g06

Time (seconds)	Optimal	HIA	BA	GA
143	6961.814	6961.781	6960.549	6929.920
180	6961.814	6961.811	6960.599	6839.698
219	6961.814	6961.796	6960.900	6833.230

IV. Conclusions and Future Works

The Human-Inspired Algorithm, a novel optimization method, is an improved way to optimize complex and continuous function. The algorithm is effective to find the global optimal or near optimal solutions since it heuristically conducts intensive search in local and global spaces based on human intelligence. The experimental results indicate that HIA obtains better solutions than BA and GA.

In the future, there are many parts of the Human-Inspired Algorithm that can be further enhanced. For example, instead of just locating one small space, a future HIA can search for many small spaces that have relatively large probabilities of finding the optimal solution. Also, there are various ways to divide a large space into smaller spaces; two factors are the shapes of the small spaces and the angles of the dividing lines or curves. The parameters in HIA can be optimized to achieve much better performance. In general, adding more human intelligence into HIA will further improve the method in optimizing complex and continuous functions.

References

[1] R. Eberhart, Y. Shi and J. Kennedy, Swarm Intelligence. Morgan Kaufmann, San Francisco, 2001.

[2] Particle Swarm Optimization, http://www.swarmintelligence.org/.

[3] D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim and M. Zaidi, "The Bees Algorithm – A Novel Tool for Complex Optimisation Problems," Technical Note, Manufacturing Engineering Centre, Cardiff University, UK., 2006.

[4] D.T. Pham, M. Castellani and A. Ghanbarzadeh, "Preliminary design using the Bees Agorithm," Proc. of Eighth International Conference on Laser Metrology, CMM and Machine Tool Performance, LAMDAMAP. Cardiff:euspan Ltd, UK, pp. 420-429, 2007.

[5] The Bees Algorithm: A Novel Tool for Optimisation Problems, http://www.bees-algorithm.com/.

[6] M. Affenzeller, A. Beham, S. Wagner and S. Winkler, "Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications," Taylor and Francis Group, Boca Raton, 2009.

[7] Genetic Algorithms, http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php.

[8] T. P. Runarsson and X. Yao, "Stochastic Ranking for Constrained Evolutionary Optimization," IEEE Transactions on Evolutionary Computation, vol. 4, no. 3, pp. 284-294, 2000.

[9] T. P. Runarsson and X. Yao, "Search Biases in Constrained Evolutionary Optimization," IEEE Transactions on Systems, Man, Cybernetics C Applications and Reviews, vol. 35, no. 2, pp. 233-243, 2005.

[10] B. Tessema and G. Yen, "An Adaptive Penalty Formulation for Constrained Evolutionary Optimization," IEEE Transactions on Systems, Man, Cybernetics A Systems and Humans, vol. 39, no. 3, pp. 565-578, 2009.