Co-evolutionary Particle Swarm Optimization to Solve min-max Problems

Yuhui Shi
EDS Embedded Systems Group
1401 E. Hoffer Street
Kokomo, IN 46902 USA
http://www.computelligence.org
Yuhui.Shi@EDS.com

Renato A. Krohling*
Departamento de Engenharia Elétrica
Universidade Federal do Espirito Santo - UFES
Av. Fernando Ferrari, CP 01-9011, CEP 29060-970
Espirito Santo, ES, Brazil
renato@ele.ufes.br

Abstract- In this paper, a co-evolutionary particle swarm optimization (PSO) to solve constrained optimization problems is proposed. First, we introduce the augmented Lagrangian to transform a constrained optimization to a min-max problem with the saddle-point solution. Next, a co-evolutionary PSO algorithm is developed with one PSO focusing on the minimum part of min-max problem while the another PSO focusing on the maximum part of the min-max problem. The two PSOs are connected through the fitness function. In the fitness calculation of one PSO, the another PSO serves as the environment to that PSO. The new algorithm is tested on three benchmark functions. The simulation results illustrate the efficiency and effectiveness of the new co-evolutionary particle swarm algorithm.

1 INTRODUCTION

In the last few years, evolutionary algorithms (EAs) have shown to be a promising approach to solve complex constrained optimization problems [1]. One of the major issues is how to handle the constraints. Some algorithms have been proposed and they can be grouped as: 1) preservation of the feasible individuals, 2) repair of infeasible solutions, 3) use of decoders, 4) penalty functions, 5) and hybrid algorithms. The performance of these methods depends on the problem at hand. The main question is to evaluate the fitness of infeasible individuals. The most employed method is the use of penalty functions.

An, at least potentially, better approach is to transform a constrained optimization problem into an unconstrained optimization problem by introducing a Lagrange multiplier. The problem then can be written in a *min-max* form, which arises in many areas of science and engineering. Min-max problems are also considered difficult to solve. Hillis [2], in his pioneering work, proposed a method inspired by the co-evolution of populations. Two independent genetic algorithms (GAs) were used with one for sorting networks (host) and the another for testing cases (parasites). Both GAs evolve simultaneously and are coupled through the fitness function. In the traditional EA, the fitness depends only on the individual of the population to be evaluated.

For co-evolutionary algorithms (Co-EA), the fitness of an individual depends on not only the individual itself but also the individuals of another EA. Generally, Co-EA can be grouped in two categories: competitive and cooperative. Although both co-evolutionary approaches have shown useful results for solving complex problems, we focus on the competitive co-evolution approach. In this case, the fitness of an individual is evaluated by means of a competition with the members of the other population [2], [3], and [4-5]. Inspired by the work of Hillis [2], Barbosa [6-7] presented a method to solve min-max problems by using two independent populations of GA coupled by the fitness function. Also, along the same line, Tahk and Sun [8] used a co-evolutionary augmented Lagrangian method to solve min-max problems by means of two populations of evolution strategies with an annealing scheme. The populations of the variable vector and the Lagrange multiplier vector approximate a zero-sum game by a static matrix game.

In this paper, inspired by the co-evolution of swarms, and based on the works of Barbosa [7] and and Tahk and Sun [8], we propose a novel method to solve min-max problems based on the co-evolution of two PSOs. Two populations of independent PSOs are evolved simultaneously: one for evolving the variable vector, and the other for evolving the Lagrange multiplier vector. The rest of the paper is organized as follows: in section 2, the min-max problem formulation is described. PSO is briefly explained in section 3. In section 4, a new approach based on co-evolutionary PSO is proposed to solve the min-max problems; Benchmark optimization problems are presented in section 5; section 6 gives simulation results with some discussions, followed by conclusions in section 7.

2 PROBLEM FORMULATION

The constrained optimization problem is expressed as

* Renato A. Krohling was sponsored by the Brazilian Research Council (CNPq) under Grant 301009/99-6

$$\min_{x\in\Re^n} f(x) \tag{1}$$

subject to

$$g_i(x)\le 0,\quad i=1,..,m$$
$$h_i(x)=0,\quad i=1,...,l$$

where the vector x consists of n variables:

$$x=[x_1,x_2,...,x_n]^T\in\Re^n$$

The set $S\subseteq\Re^n$ designates the search space, which is defined by the lower and upper bounds of the variables: $x_{il}\le x_i\le x_{iu}$ with $i=1,...,n$. By introducing the Lagrangian formulation, the dual problem associated with the primal problem (1) can be written as

$$\max_{\lambda,\mu} L(x,\mu,\lambda) \tag{2}$$

subject to

$$\mu_i\ge 0,\qquad \text{for } i=1,...,m$$

with

$$L(x,\mu,\lambda)=f(x)+\mu^T g(x)+\lambda^T h(x) \tag{3}$$

and μ is a $m\times 1$ multiplier vector for the inequality constraints and λ is a $l\times 1$ multiplier for the equality constraints. If the problem (1) satisfies the convexity conditions over S, then the solution of the primal problem (1) is the vector x^* of the saddle-point $\{x^*,\lambda^*,\mu^*\}$ of $L(x,\lambda,\mu)$ so that:

$$L(x^*,\mu,\lambda)\le L(x^*,\mu^*,\lambda^*)\le L(x,\mu^*,\lambda^*)$$

Solving the min-max problem

$$\min_x \max_{\lambda,\mu} L(x,\mu,\lambda) \tag{4}$$

provides the minimizer x^* as well as the multiplier μ^*,λ^*.

For non-convex problems, however, the solution of the dual problem does not coincide with that of the primal problem. In this case, to the Lagrangian function is added a penalty function

$$P(x)=\frac{P}{2}\left(\sum_{i=1}^{m}(g_i^{2+}(x)+\sum_{i=1}^{l}(h_i^2(x))\right) \tag{5}$$

where P is a positive penalty parameter, and

$$g_i^+(x)=\max(0,g_i(x)),\quad \text{for } i=1,...,m$$

The augmented Lagrangian is written then as

$$L_a(x,\mu,\lambda,p)=f(x)+\mu^T(x)+\lambda^T h(x)+P(x)$$

A very usually form to describe the augmented Lagrangian is as following [8]:

$$L_a(x,\mu,\lambda,\rho)=f(x)+\sum_{k=1}^{m}p_k(x,\mu,\rho)+$$
$$+\lambda^T h(x)+\rho\sum_{k=1}^{l}(h_k^2(x)) \tag{6}$$

where the term p_k for the k-th inequality constraint is given by

$$p_k(x,\mu_k,\rho)=\begin{cases}(\mu_i g_i(x)+\rho g_i^2(x), & \text{if } g_i(x)\ge\frac{-\mu_i}{2\rho}\\ \frac{-\mu_i^2}{4\rho} & \text{if } g_i(x)<\frac{-\mu_i}{2\rho}\end{cases} \tag{7}$$

It can be demonstrated that the solution of the primal problem and the augmented Lagrangian are identical [9]. The main issue is how to find the saddle-point $\{x^*,\lambda^*,\mu^*\}$. In Section 4, a Co-PSO based on the evolution of both the variable vector and the Lagrangian multiplier is proposed to solve the min-max problem.

3 PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (*PSO*) is an evolutionary computation technique originally developed by Kennedy and Eberhart [11], Eberhart and Kennedy [10]. The *PSO* is motivated from the simulation of social behavior instead of the evolution of nature as in the other evolutionary algorithms (genetic algorithms, evolutionary programming, evolutionary strategies, and genetic programming). It is a population-based evolutionary algorithm. Similar to the other population-based evolutionary algorithms, *PSO* is initialized with a population of random solutions. Unlike the most of the evolutionary algorithms, each potential solution (individual) in *PSO* is also associated with a randomized velocity, and the potential solutions, called *particles*, are then "flown" through the problem space [15].

Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far. This value is called *pbest*. Another "*best*" value that is tracked by the *global* version of the particle swarm optimizer is the overall best value, and its

location, obtained so far by any particle in the population. This location is called *gbest*.

The particle swarm optimization concept consists of, at each time step, changing the velocity (accelerating) of each particle flying toward its *pbest* and *gbest* locations (global version of *PSO*). Acceleration is weighted by random terms, with separate random numbers being generated for acceleration toward *pbest* and *gbest* locations, respectively. The procedure for implementing the global version of *PSO* is is given in the List 1.

List 1: Procedure of PSO

1) Initialize a population (array) of particles with random positions and velocities in the n dimensional problem space.
2) For each particle, evaluate its fitness value.
3) Compare each particle's fitness evaluation with the particle's *pbest*. If current value is better than *pbest*, then set *pbest* value equal to the current value and the *pbest* location equal to the current location in n-dimensional space.
4) Compare fitness evaluation with the population's overall previous best. If current value is better than *gbest*, then reset *gbest* to the current particle's array index and value.
5) Change the velocity and position of the particle according to equations (8) and (9) [12-13]:

$$\boldsymbol{v}_i = w\boldsymbol{v}_i + c_1\, rand()(\boldsymbol{p}_i - \boldsymbol{x}_i) + c_2\, Rand()\,(\boldsymbol{p}_g - \boldsymbol{v}_i) \tag{8}$$

$$\boldsymbol{x}_i = (\boldsymbol{x}_i + \boldsymbol{v}_i) \tag{9}$$

6) Loop to step 2) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations (generations).

where $\boldsymbol{x}_i = [x_{i1}, x_{i2}, ..., x_{in}]^T$ stands for the position of the i-th particle, $\boldsymbol{v}_i = [v_{i1}, v_{i2}, ..., v_{in}]^T$ stands for the velocity of the i-th particle and $\boldsymbol{p}_i = [p_{i1}, p_{i2}, ..., p_{in}]^T$ represents the best previous position (the position giving the best fitness value) of the i-th particle. The index g represents the index of the best particle among all the particles in the group. Variable w is the inertia weight, c_1 and c_2 are positive constants; *rand*() and *Rand*() are two random functions in the range [0, 1]. Particles' velocities on each dimension are clamped to a maximum velocity *Vmax*. If the sum of accelerations would cause the velocity on that dimension to exceed *Vmax*, which is a parameter specified by the user, then the velocity on that dimension is limited to *Vmax*.

Vmax is an important parameter. It determines the resolution with which the regions around the current solutions are searched. If *Vmax* is too high, the *PSO* facilitates global search, and particles might fly past good solutions. If *Vmax* is too small, on the other hand, the *PSO* facilitates local search, and particles may not explore sufficiently beyond locally good regions. In fact, they could become trapped in local optima, unable to move far enough to reach a better position in the problem space [14].

The first part in equation (8) is the momentum part of the particle. The inertia weight w represents the degree of the momentum of the particles. The second part is the "cognition" part, which represents the independent thinking of the particle itself. The third part is the "social" part, which represents the collaboration among the particles. The constants c_1 and c_2 represent the weighting of the "cognition" and "social" parts that pull each particle toward *pbest* and *gbest* positions. Thus, adjustment of these constants changes the amount of "tension" in the system. Low values allow particles to roam far from already found better regions before being tugged back, while high values result in abrupt movement toward, or past, already found better regions.

Early experience with particle swarm optimization (trial and error, mostly) led us to set the acceleration constants c_1 and c_2 equal to 2.0 for almost all applications. *Vmax* is often set at about 10-20% of the dynamic range of the variable on each dimension.

The population size selected was problem-dependent. Population sizes of 20-50 were probably most common. It is learned that small population sizes are acceptable for *PSO* to be optimal in terms of minimizing the total number of evaluations (population size times the number of generations) needed to obtain a sufficient solution.

4 CO-EVOLUTIONARY PSO

Two populations of PSOs are involved in the co-evolutionary PSO to solve the min-max problem formulated in the expression (4). The first PSO focuses on evolving the variable vector $\boldsymbol{x}$ with "frozen" $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$. Only the variable vector $\boldsymbol{x}$ is represented in the population P_1 The second PSO focuses on evolving Lagrangian multiplier vectors $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ with "frozen" $\boldsymbol{x}$. Only the multiplier vectors $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are represented in the population P_2. The two PSOs interact with each other through the fitness evaluation. For the first PSO, the problem is a minimum problem and the fitness value of each individual $\boldsymbol{x}$ is evaluated according to

$$f(\boldsymbol{x}) = \max_{\boldsymbol{\mu},\boldsymbol{\lambda} \in P_2} L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) \tag{10}$$

For the second problem, the problem is a maximum problem and the fitness value of each individual $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ is evaluated according to

$$g(\mu,\lambda) = \min_{x \in P_1} L(x,\mu,\lambda) \qquad (11)$$

Since in the PSO algorithm, all the particles (or individuals) survive into the next generation, there is no selection mechanism involved. The cooperation among the particles are through the "history" *pbest* and *gbest*, which are updated if better fitness values are obtained so the PSO can be applied to solve both minimum and maximum problems.

The procedure of the co-evolutionary PSO algorithm is given in the List 2. Within each cycle, the first PSO is run for max_gen_1 generations, then the second PSO is run for max_gen_2 generations, this process is repeated until either an acceptable solution has been obtained or the maximum number of cycles has been reached. The global best in the population P_1 is the solution for the variable vector x, and the global best in the population P_2 is the solution for the Lagrangian multiplier vectors μ and λ.

List 2: Procedure of co-evolutionary PSO

1) Initialize two PSOs.
2) Run the first PSO for max_gen_1 generations.
3) Re-evaluate the *pbest* values for the second PSO if it is not the first cycle.
4) Run the second PSO for max_gen_2 generations.
5) Re-evaluate the *pbest* values for the first PSO.
6) Loop to step 2 until a termination condition is met.

In the above co-evolutionary PSO algorithm, when one PSO is running, the other serves as its environment, so each PSO has a changing environment from cycle to cycle. Due to the change of the environment between cycles, the *pbest* values obtained in the previous cycle has to be re-evaluated according to the new environment before starting its evolving, which is shown as step 3 and 5 in the List 2.

5 EXPERIMENTAL SETTING

For comparison, three benchmark constrained optimization problems reported in [1] and [8] are used here. The first optimization problem G1 consists of minimizing:

$$f(x) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

subject to

$$\begin{aligned}
&2x_1 + 2x_2 + x_{10} + x_{11} \le 10\\
&2x_1 + 2x_3 + x_{10} + x_{12} \le 10\\
&2x_2 + 2x_3 + x_{11} + x_{12} \le 10\\
&-8x_1 + x_{10} \le 0\\
&-8x_2 + x_{11} \le 0\\
&-8x_3 + x_{12} \le 0\\
&-2x_4 - x_5 + x_{10} \le 0\\
&-2x_6 - x_7 + x_{11} \le 0\\
&-2x_8 - x_9 + x_{12} \le 0
\end{aligned}$$

where

$$\begin{aligned}
&0 \le x_i \le 1, && i = 1,...,9;\\
&0 \le x_i \le 100, && i = 10,11,12;\\
&0 \le x_i \le 1, && i = 13.
\end{aligned}$$

The global minimum is known to be

$$x^* = (1,1,1,1,1,1,1,1,1,3,3,3,1)$$

with $f(x^*) = -15$.

The second optimization problem G7 consists of minimizing:

$$\begin{aligned}
f(x) = {} & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2\\
& + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2\\
& + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45
\end{aligned}$$

subject to

$$\begin{aligned}
&105 - 4x_1 - 5x_2 + 3x_7 - 9x_9 \ge 0\\
&-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \ge 0\\
&-10x_1 + 8x_2 + 17x_7 - 2x_8 \ge 0\\
&-x_1^2 - 2x(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \ge 0\\
&8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \ge 0\\
&-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \ge 0\\
&3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \ge 0\\
&-0.5(x_1 - 8)^2 - 2(x_2 - 4) - 3x_5^2 + x_6 + 30 \ge 0
\end{aligned}$$

where

$$-10 \le x_i \le 10, \qquad i = 1,\cdots,10.$$

The global minimum is known to be

$$\begin{aligned}
x^* = {} & (2.171996, 2.363683, 8.773926, 5.095984,\\
& 0.9906548, 1.430574, 1.321644, 9.828726,\\
& 8.280092, 8.375927)
\end{aligned}$$

with $f(x^*) = 24.3062091$.

The last optimization problem G9 consists of minimizing:

$$\begin{aligned}
f(x) = {} & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2\\
& + 10x_5^6 + 7x_6^2 - 4x_6x_7 - 10x_6 - 8x_7
\end{aligned}$$

subject to

$$\begin{aligned}
&127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \ge 0\\
&282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \ge 0\\
&196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \ge 0\\
&-4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \ge 0
\end{aligned}$$

where

$$-10 \le x_i \le 10, \qquad i = 1, \cdots, 7.$$

The global minimum is known to be

$$x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$$

with $f(x^*) = 680.6300573$.

For all three benchmark problems, the population sizes are set as 40 and 30, respectively. The maximum number of generations for each PSO of one cycle is chosen to be 10. To test the convergence speed of the co-evolutionary PSO, three different maximum numbers of cycles are chosen: 40, 80 and 120. The particles are randomly initialized within the boundaries for each run. The inertia weight of each PSO is linearly decreased over the course of each run, starting from 0.9 and ending at 0.4. Each different parameter setting is run 50 times. Each run is terminated only when the maximum number of cycles has been reached.

6 EXPERIMENTAL RESULTS AND DISCUSSION

Table 1, 2, and 3 list the simulation results for the three benchmark problems averaged over 50 runs, respectively.

Table 1: Results for G1 over 50 runs

Max. Number Cycles	Function minimum values	
	Average	variance
40	-13.9637	1.094654
80	-14.0373	1.119829
120	-14.1597	1.092294

Table 2: Results for G7 over 50 runs

Max. Number Cycles	Function minimum values	
	Average	variance
40	24.70024076	0.212047124
80	24.5433936	0.057324594
120	24.47762322	0.039600375

Table 3: Results for G9 over 50 runs

Max. Number Cycles	Function minimum values	
	Average	variance
40	680.8729233	0.02885334
80	680.7091197	0.004666228
120	680.668411	0.001000557

Table 4: First 20 run results for G1 with maximum number of cycles 40

-12.9566	-14.9441	-14.9668	-14.9646
-14.9514	-12.9704	-12.9752	-12.9557
-12.4198	-14.9478	-12.9554	-12.9834
-12.9481	-12.9562	-14.9492	-14.9532
-14.9604	-14.9426	-14.9486	-14.9577

From Table 1, 2 and 3, it can be easily seen that the co-evolutionary PSO can converge very quickly towards the global true optimum, except for G1 problem. For G1 problem, it has big variance over the 50 runs under all three different maximum numbers of cycles. By looking at the results for all 50 runs, it can be seen that the co-evolutionary PSO finds solution either at value between 12 and 13 or at value between 14 and 15. The results for first 20 runs are listed in Table 4 for illustration. To have better results, at least for G1 problem, the co-evolutionary PSO algorithm needs to be modified in some way to avoid the algorithm to trap at some local attractors.

7 CONCLUSIONS

In this paper, inspired by the co-evolution of swarms, a co-evolutionary PSO has been developed to solve min-max problems, and tested on three benchmark problems. The simulation results illuminate that the new algorithm, for most cases, can quickly search towards the global optimum. More works need to be done to improve the new algorithm's ability to escape the local attractors, and to demonstrate the algorithm's ability to fine tune the solution.

REFERENCES

[1] Michalewicz, Z. and Schoenauer, M., Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation,* Vol. 4, No. 1, pp. 1-32, 1996.

[2] Hillis, W.D., Coevolving parasites improve simulated evolution as an optimization procedure. *Physica D*, Vol. 42, pp. 228-234, 1990.

[3] Paredis, J., Steps towards coevolutionary classification neural networks, Artificial Life IV, MIT Press, pp. 359-365, 1994.

[4] Rosin, C.D., and Belew, R.K., Methods for competitive coevolution: Finding opponents worth beating. *Proc. of the 6th int. Conf. on Genetic Algorithms and their Applications.* Pittsburgh, USA, pp. 373-380, 1995.

[5] Rosin, C.D. and Belew, R.K., New methods for competitive coevolution. *Evolutionary Computation,* Vol. 5, No. 1, pp. 1-29, 1996.

[6] Barbosa, H.J.C., A genetic algorithm for min-max problems. *Proc. of the 1st int. conf. on Evolutionary Computation and its Applications,* Moscow, Russia, pp. 99-109, 1996.

[7] Barbosa, H.J.C., A coevolutionary genetic algorithm for constrained optimization. *Proc. of the 1999 Congress on Evolutionary Computation*, pp. 1605-1611, 1999.

[8] Tahk, M.J. and Sun, B.-C., Coevolutionary augmented lagrangian methods for constrained optimization. *IEEE Trans. on Evolutionary Computation,* Vol. 4, No. 2, pp. 114-124, 2000.

[9] Bazaraa, M.S., Sherali, H.D. and Shetty, C.M., Nonlinear Programming: Theory and Algorithms, 2nd. Ed. New York: Wiley, 1993.

[10] Eberhart, R.C. and Kennedy, J., A new optimizer using particle swarm theory. *Proc. of the 6th. Int. Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp. 39-43. Piscataway, NJ: IEEE Service Center, 1995.

[11] Kennedy, J. and Eberhart, R.C., Particle swarm optimization. *Proc. of the IEEE Int. Conf. on Neural Networks IV*, Piscataway, NJ: IEEE Service Center, pp.1942–1948, 1995.

[12] Shi, Y. and Eberhart, R.C., Parameter selection in particle swarm optimization. In *Evolutionary Programming VII: Proc. EP98*, New York: Springer-Verlag, pp. 591-600, 1998.

[13] Shi, Y. and Eberhart, R.C., A modified particle swarm optimizer. *Proc. of the IEEE International Conference on Evolutionary Computation,* Piscataway, NJ: IEEE Press, pp. 69-73, 1998.

[14] Fan, H.-Y. and Shi, Y., Study of Vmax of the particle swarm optimization algorithm. *Proc. of the Workshop on Particle Swarm Optimization*. Indianapolis, IN: Purdue School of Engineering and Technology, IUPUI, 2001.

[15] Kennedy, J., Eberhart, R.C. and Shi, Y., *Swarm Intelligence*, San Francisco: Morgan Kaufmann Publishers, 2001.