Globally Convergent Particle Swarm Optimization via Branch-and-Bound

Zaiyong Tang (Corresponding author)
Dept. of Marketing and Decision Sciences, Salem State University
Salem, MA 01970, USA
E-mail: ztang@salemstate.edu

Kallol Kumar Bagchi
Dept. of Information & Decision Sciences, University of Texas at El Paso
El Paso, TX 79968-0544, USA

Abstract

Particle swarm optimization (PSO) is a recently developed optimization method that has attracted interest of researchers in various areas. PSO has been shown to be effective in solving a variety of complex optimization problems. With properly chosen parameters, PSO can converge to local optima. However, conventional PSO does not have global convergence. Empirical evidences indicate that the PSO algorithm may fail to reach global optimal solutions for complex problems. We propose to combine the branch-and-bound framework with the particle swarm optimization algorithm. With this integrated approach, convergence to global optimal solutions is theoretically guaranteed. We have developed and implemented the BB-PSO algorithm that combines the efficiency of PSO and effectiveness of the branch-and-bound method. The BB-PSO method was tested with a set of standard benchmark optimization problems. Experimental results confirm that BB-PSO is effective in finding global optimal solutions to problems that may cause difficulties for the PSO algorithm.

Keywords: Particle swarm optimization, Global optimal solution, Branch-and-bound, Hybrid method

1. Introduction

Particle swarm optimization (PSO) is a global optimization technique developed by Eberhart and Kennedy (1995). Each particle in a PSO system represents a feasible solution in the solution space of the problem. Since particles "fly" in the search space in a way similar to the social behavior of bird flocking or fish schooling, PSO is often referred to as an evolutionary computation method. Compared with other evolutionary techniques such as genetic algorithms (Goldberg, 1989) and evolutionary programming (Fogel, 1994), PSO is relatively easy to implement and there are few parameters to adjust in the algorithm. Recently published results have shown that PSO can be more effective in solving complex optimization problems than traditional optimization approaches. PSO has been applied to a wide variety of global optimization problems, including both unconstrained (Eberhart, Simpson, and Dobbins, 1996; Angeline, 1998; Kennedy & Eberhart 2001) and constrained optimization problems (Hu, Eberhart, & Shi, 2003). Barrera and Coello (2009) have successfully applied PSO to solve multi-modal optimization problems. Padhye, Branke, and Mostaghim (2009) developed a PSO algorithms aimed at solving multi-objective optimization problems. Chena, et al. (2010) proposeed an improved particle swam optimization approach to solve resource-constrained scheduling problem. Venter and Haftka (2010) converted single objective optimization problem into bi-objective optimization problem solved by a multi-objective implementation of particle swarm optimization. Banks, Vincet, and Anyakoha (2007) presented an extensive review of research in PSO.

Although PSO has been used successfully in solving many difficult optimization problems, it does not guarantee convergence to a global optimal solution in the limit. Van den Bergh (2002) showed that the original PSO algorithm (Eberhart & Kennedy, 1995) may converge prematurely to suboptimal solutions. In practice, satisfactory solutions may be found with properly chosen parameters of the PSO algorithm. However, a guaranteed global convergence is needed for some critical applications, in which case the current form of PSO cannot be trusted. Furthermore, for problems where the global optima are unknown, questions such as how good the current solution is when the program terminates are left unanswered.

Recently, Clerc and Kennedy (2002) studied the convergence of PSO in multi-dimensional complex search spaces. They proposed a generalized model with methods for controlling the convergence properties of PSO systems. They showed that with appropriately chosen parameters, convergence could be achieved. Van den

Bergh (2002) developed a new PSO algorithm called the Guaranteed Convergence Particle Swarm Optimizer (GCPSO) which is proven to converge onto local minima. However, neither the Clerc and Kennedy algorithm nor the GCPSO guarantees the convergence to global optimal solutions.

In this paper, we propose a branch-and-bound based method to render guaranteed convergence to global optimal solutions for PSO. In the following sections, we first briefly discuss the PSO algorithm. Then we will present the branch-and-bound framework developed by Horst and Tuy (1990). A brief discussion of global convergence follows. Next, we discuss the implementation of a combined branch-and-bound and particle swarm optimization (BB-PSO) approach, followed by experiments of testing the proposed methods with a set of standard benchmark problems. We conclude with a discussion of the experimental results and further research issues, and a brief summary.

2. Particle Swarm Optimization

In a PSO system, a large number of particles are deployed to "fly" through the solution space of a given problem. Each particle represents a feasible solution for the problem. The search of a new solution by each particle is motivated by the idea of social learning, in which individuals learn from their own experience as well as the experiences of those in their social network (neighborhood). Thus, each particle determines its next move by making use of its own best position achieved so far and the current best position of its neighbors. The quality of a solution is measured according to a fitness function. Note that for minimization problems, an error function is used in lieu of the fitness function.

Algorithm PSO

1) Initialize: set iteration number $t=0$.

Set the swarm domain D (search space). Start with a swarm, S, of randomly distributed particles $\boldsymbol{x}$ and velocity $\boldsymbol{v}$ of dimension d.

2) Evaluate: at iteration $t=t+1$.

For each $\boldsymbol{x}_p \in S$, find the criterion function value $f(\boldsymbol{x}_p)$.

If $f(\boldsymbol{x}_p^*) \le$ Error_tolerance, where $\boldsymbol{x}_p^* = \text{argmin } \{f(\boldsymbol{x}_p) \mid \forall \boldsymbol{x}_p \in S\}$ or $t >$ Max_iteration, then stop. $\boldsymbol{x}_p^*$ is a global optimal solution. Otherwise, go to Step 3.

3) Update:

For each $\boldsymbol{x}_p \in S$, find $\boldsymbol{n}_p$, the best neighbor of $\boldsymbol{x}_p$. $\boldsymbol{n}_p = \text{argmin } \{f(\boldsymbol{x}_p) \mid \text{all } \forall \boldsymbol{x}_p \in N(\boldsymbol{x}_p)\}$. $N(\boldsymbol{x}_p)$ is $\boldsymbol{x}_p$'s neighborhood. The velocity of particle p is updated using:

$$\boldsymbol{v}_p = \boldsymbol{v}_p + rand()*(\boldsymbol{b}_p - \boldsymbol{x}_p) + rand()*(\boldsymbol{n}_p - \boldsymbol{x}_p)$$

where $\boldsymbol{b}_p$ is the current best solution of particle p achieved over iterations $0,1, \ldots t-1$. $rand()$ returns a uniform random number between 0 and 1. The position of particle p is updated using:

$$\boldsymbol{x}_p = \boldsymbol{x}_p + \boldsymbol{v}_p.$$

Go to Step 2.

Remarks:

1) This is the basic PSO algorithm. There are a number of variations aimed at improving the performance of the algorithm (Kennedy & Eberhart, 2001).

2) In order to avoid the divergence of the particle trajectories, $\boldsymbol{v}$ is capped by a predetermined maximum value. That is, $\boldsymbol{v}_p \le$ Max_velocity. Clamping with maximum velocity is optional. Van den Bergh and Engelbrecht (2002).have found in some problems that no clamping should be used.

3) In our implementation, we followed Shi and Eberhart (1998a) to use an inertia weight w to adjust the convergence behavior of the algorithm.

$$\boldsymbol{v}_p = w*\boldsymbol{v}_p + rand()*(\boldsymbol{b}_p - \boldsymbol{x}_p) + rand()*(\boldsymbol{n}_p - \boldsymbol{x}_p)$$

Shi and Eberhart (1998b) have shown that PSO's performance may be significantly influenced by the choice of w. Small values of w leads to more rapid convergence, but potentially suboptimal solutions (Van den Bergh & Engelbrecht, 2002). In general, Large weight values will favor exploration, while small values favor exploitation.

4) $N(\boldsymbol{x}_p)$ can be expanded to the whole domain D. Then $\boldsymbol{n}_p$ is replace by $\boldsymbol{g}_p$ in step 3, where $\boldsymbol{g}_p = \text{argmin } \{f(\boldsymbol{x}_p) \mid \text{all } \forall \boldsymbol{x}_p \in D\}$.

PSO and several of its variations have been shown to be effective in solving a wide range of complex problems. Empirical evidence (e.g., Shi & Eberhart, 1998b) shows that PSO may fail to reach a solution within the maximum number of iterations. The likelihood of such events depends on the selection of parameters such as the inertia weight and the maximum velocity allowed. Recent work by Clerc and Kennedy (2002) and Van den Bergh and Engelbrecht (2002) has addressed the local convergence issue of PSO algorithms. However, for many real world problems, we need insurance that global optima are found. This need for global optima motivated us to consider a branch-and-bound based global optimization approach to give PSO guaranteed convergence to global optimal solutions.

3. Branch and Bound

Branch-and-Bound (BB) is a general class of global optimization algorithms that work in a finite domain. In general, a BB algorithm has a branching rule that divides the search space into multiple partitions; a bounding rule that provides a lower bound and an upper bound on the value of each solution in a partition generated by the branching rule; and a search strategy that determines the next partition from which to branch. The branching and bounding process continues until the lower bound matches the upper bound, which happens when a global optimum is found. Note that for minimization problems, when the lower bound for a partition is higher than a known solution (upper bound), that partition is eliminated from further consideration as it cannot contain a global optimal solution. In practice, rather than requiring the lower bound and the upper bound to be equal for the algorithm to stop, the algorithm terminates when the difference between the two bounds is sufficiently small.

Horst (1986) developed a general theoretic framework for solving deterministic global optimization problems using branch-and-bound algorithms. Implementation of the general BB algorithms typically requires taking into account the specific features of the problems to be solved. In the following, we first introduce the definition of partition in a BB algorithm. Then we present the general algorithm and related convergence theorems.

Partition: Let M be a closed set in R^S and let I be a finite index set. A set $\mathrm{M} = \{M_i \mid i \in I\}$ of closed subsets of M is said to be a partition of M if

$$M = \bigcup_{i \in I} M_i$$

and

$$M_i \bigcup M_j = \partial M_i \bigcup \partial M_j, \ \forall i, j \in I, i \neq j$$

where ∂M_i denotes the (relative) boundary of M_i.

Let $F : D \in R^S \rightarrow R$ be the global function to be minimized, and let $M_i \in$ M. We say M_i is feasible if $M_i \bigcap D \neq \varnothing$, and M_i is infeasible if $M_i \bigcap D = \varnothing$. Otherwise M_i is uncertain. A subset is active if it is feasible or uncertain.

Let n be the iteration index of the BB algorithm. We use M_n to denote the collection of active subsets, and α_n, β_n to denote the upper and lower bound of F, respectively, at iteration n. Following the formalism of Horst and Tuy (1990, p.114) we rewrite the general BB algorithm below. The convention that infima and minima taken over an empty set equal to $+\infty$ is observed.

Algorithm BB

1) Initialization:

a. Set $n = 0$.

b. Choose a relaxed feasible set $M_0 \supseteq D$, and a possibly empty feasible set $S_{M_0} \subset D$. Set $\mathrm{M}_0 = \{M_0\}$, and find upper and lower bounds associated with M_0. That is, $\alpha_0 = \alpha(M_0)$, $\beta_0 = \beta(M_0)$, satisfying

$$\beta_0 \leq \min F(D) \leq \alpha_0 = \min F(S_{M_0}).$$

c. If $\alpha_0 < \infty$, then choose the current best solution $\boldsymbol{x}^0$ such that $F(\boldsymbol{x}^0) = \alpha_0$.

d. If $\alpha_0 - \beta_0 = 0$, then stop. $\boldsymbol{x}^0$ is a global optimal solution. Otherwise, go to Step 2.

2) Recursion:

e. Set $n = n - 1$. At the beginning of iteration n, the current partition M_{n-1} contains all the active subsets $\{M_i \mid i \in I_{n-1}\}$.

f.For each M_i, $i \in I_{n-1}$, we have upper and lower bounds satisfying:

$$\beta(M_i) \le \min F(M_i \bigcap D) \le \alpha(M_i) = \min F(S_{M_i})$$

if M_i is feasible and

$$\beta(M_i) \le \min F(M_i)$$

if M_i is uncertain.

g. Compute overall upper and lower bounds α_{n-1}, β_{n-1} satisfying

$\alpha_{n-1} = \min \alpha(M_i)$ for all $M_i \in M_{n-1}$;

$\beta_{n-1} = \min \beta(M_i)$ for all $M_i \in M_{n-1}$; and

$\beta_{n-1} \le \min F(D) \le \alpha_{n-1}$.

2.1 Branching:

a. Let R_n be the collection of all subsets $M_i \in M_{n-1}$ such that $\beta(M_i) \le \alpha_{n-1}$, i.e., retaining only subsets that are still of interest.

b. Select a nonempty collection of sets $P_n \subset R_n$ and partition each member of P_n. Let P_n^{new} be the collection of all the newly formed subsets.

c. Let M_n^{new} be the collection of all the subsets $M_i \in P_n^{new}$ for all active M_i. (Feasible or uncertain).

2.2 Bounding:

a. For each $M_i \in M_n^{new}$, find:

$S_{M_i} \in M_i \bigcap D$ (if possible);

$\beta(M_i) = \min F(M_i \bigcap D)$ if M_i is feasible;

$\beta(M_i) = \min F(M_i)$ if M_i is uncertain; and

$\alpha(M_i) = \min F(S_{M_i})$.

b. Set $M_n = (R_n \setminus P_n) \bigcup M_n^{new}$, i.e., merge all subsets still of interest. Let

$\alpha_n = \min \alpha(M_i)$ for all $M_i \in M_n$ and

$\beta_n = \min \beta(M_i)$ for all $M_i \in M_n$.

c. Update the current solution:

If $\alpha_n < \infty$, let $\boldsymbol{x}^n \in D$ such that $F(\boldsymbol{x}^n) = \alpha_n$.

3) Stopping Condition:

If $\alpha_n - \beta_n = 0$, then stop. $\alpha_n = \beta_n = \min F(D)$. $\boldsymbol{x}^n$ is a global optimal solution. Otherwise, go to Step 2.

Remarks:

1) The general BB algorithm leaves many implementation details to be determined by specific applications. The efficiency and convergence of the BB procedure depends on three important questions: (1) how to carry out the partition; (2) how to determine the bounds (bounding); and (3) how to choose the collection of subsets for further partitioning (branching).

2) The stopping condition $\alpha_n - \beta_n = 0$ can be relaxed to $\alpha_n - \beta_n \le \varepsilon$, where $\varepsilon \in R^+$ is small. Conventionally, ε is referred to as the error tolerance.

3) Since $\{\alpha_n\}$ is non-increasing and $\{\beta_n\}$ is non-decreasing, the limit $\alpha = \lim_{n\to\infty} \alpha_n$ and the limit $\beta = \lim_{n\to\infty} \beta_n$ exist. Also $\beta \le \min F(D) \le \alpha$ by the construction of the algorithm.

4) In general, obtaining upper bounds is much easier than obtaining lower bounds, as the former can be any feasible solution. For implementing the BB to particle swarm optimization, the upper bound is given by the current global best solution at each iteration, while the lower bound is the known global minimum if it is available. If the global minima are unknown, methods that exploit the intrinsic properties of the criterion function, such as the Piyavskii algorithm (Piyavskii, 1972), can be used to obtain lower bounds.

5) Many practical problems have known theoretical minimum criterion values that can be used. For example, the training of neural networks has a theoretical minimum sum-of-squares error of zero.

In order to establish convergence results for BB based algorithms, several definitions and convergence theorems due to Horst and Tuy (1990) are presented below.

Consistent Bounding: a bounding operation is consistent if at every iteration n any active partition element can be further refined, and if any decreasing sequence $\{M_{n_q}\}$ of successively refined partition elements satisfies

$$\mathrm{Lim}_{q\to\infty}(\alpha_{n_q} - \beta(M_{n_q})) = 0.$$

If $\{M_{n_q}\}$ is finite, then the bounding operation is called finitely consistent.

Complete Branching: A branching operation is complete if in the end, for every feasible partition element $M \in \bigcup_{p=1}^{\infty}\bigcap_{n=p}^{\infty} \mathrm{R}_n$, we have

$$\min \mathrm{F}(M \bigcap D) \geq \alpha \equiv \lim_{n\to\infty} \alpha_n .$$

That is, any unexplored elements of feasible partitions cannot contain a better solution.

Bound Improving Branching: A branching operation is bound improving if, at least each time after a finite number of iterations, one of the partition elements with the best lower bound is selected for further partition. This requires

$$\mathrm{P}_n \bigcap \mathrm{argmin}\{\beta(M_i) \mid M_i \in \mathrm{R}_n\} \neq \varnothing.$$

Note that if the bounding operation is consistent, then a bound improving branching operation is also complete. The following two theorems provide the convergence of the BB algorithm (see Horst & Tuy, 1990 for proofs):

Theorem 1. In an infinite branch and bound procedure, suppose the bounding operation is consistent and the branching operation is complete, then

$$\alpha \equiv \lim_{n\to\infty} \alpha_n = \lim_{n\to\infty} F(\boldsymbol{x}^n) = \min \mathrm{F}(D).$$

Theorem 2. In an infinite branch and bound procedure, suppose the bounding operation is consistent and branching operation is bound improving, then the procedure is convergent. We have

$$\alpha \equiv \lim_{n\to\infty} \alpha_n = \lim_{n\to\infty} F(\boldsymbol{x}^n) = \min \mathrm{F}(D) = \lim_{n\to\infty} \beta_n \equiv \beta.$$

4. Combining BB and PSO

There are two practical ways of combining BB and PSO. The first method is embedding BB within a PSO. In this case, the particle population can be divided into sub-populations, or clusters. Each sub-population is confined to a partition of the search space defined in the BB algorithm. This method of distributing particles in the partitions is analogous to stratified sampling in which a better coverage of the entire search space is achieved. The particles are allowed to traverse across partition boundaries, congregating to partitions that are more promising in the possibility of containing optimal solutions. The branching operation selects a partition based on a predetermined search criterion (such as best solution first, depth first, or breadth first), and the bounding operation updates the upper bound of each partition using the PSO algorithm.

The second approach is embedding PSO within the BB framework. Each partition element has its own particle swarm, and PSO is carried out within the confinement of the current partition of BB. At the initial step, the whole search space is one partition. If an acceptable solution is found, then no branch and bound procedure is invoked. When the PSO algorithm fails to reach an acceptable solution, then the BB process starts. The second approach is easier than the first to implement, as the PSO algorithm needs not to be modified, other than that the particle search space is re-defined based on the current partition in the BB algorithm. Using the second approach to combine BB and PSO, we discuss the implementation and convergence property of the combined method in the following.

Partition the search space

Without loss of generality, we can assume that the search space is a hyper-cube D with dimension d for unconstrained optimization problems ($D \subset R^d$). In each iteration of the branch-and-bound procedure, branching involves first selecting a candidate partition and then cutting the hyper-cube with d -1 dimension hyper-planes. If we choose the hyper-planes in such a way that each is perpendicular to one axis and cut the hyper-cube along a fixed point of the edges, then there are d such hyper-planes. Applying the d hyper-planes to D will result in 2^d new partitions. Figure 1 shows the case when $d = 3$ and the division occurs at the middle point between max x_3 and min x_3. For simplicity, it is generally desirable to cut the hyper-cube at the middle point of the edges.

(Insert Figure 1 here)

This partition scheme becomes very computationally intensive for problems with a large dimension *d* as the number of new partitions increases exponentially. A practical implementation of partitioning is to apply only one cut to the current partition at each branching operation, resulting in two new partitions regardless of the size of dimension *d*.

Pseudo Algorithm of BB-PSO

1) Start BB: set initial partition.

2) Apply PSO to the initial partition. If a solution is found, stop; otherwise, continue.

3) Branching operation: select an active partition and divide it to smaller partitions.

4) Bounding operation: use PSO to obtain upper bounds and use a known global minimum $\boldsymbol{x}^t$ for computing lower bounds. Find the current best solution $\boldsymbol{x}^*$ among all active partitions.

5) Evaluation: If $|f(\boldsymbol{x}^*) - f(\boldsymbol{x}^t)| \leq$ Error_tolerance, or iteration > Max_iteration, then stop. $\boldsymbol{x}^*$ is a global optimal solution. Otherwise, go to Step 3.

Remarks:

1) Any feasible solution in a partition is an upper bound of the optimal solution in that partition. The closer an upper bound is to the lower bound or the known optima, the better the upper bound is. The global best solution from the PSO algorithm applied within each partition is the upper bound of solutions in the partition.

2) Finding lower bounds is much more difficult in general. Problem specific knowledge is needed to design effective methods for computing lower bounds for the partitions. For instance, when the fitness function is Lipschitz continuous, the Lipschitz constant of the function can be used to estimate the lower bounds in each partition. When the global optima are known for a given problem, they become the best lower bounds as they are the targets of the successive sequence of upper bounds.

Convergence of BB-PSO

When the search space *D* is bounded, the partition scheme discussed above can be applied successively to each and every new partition in the BB procedure. The partition scheme is exhaustive, that is, any active partition element can be further refined and no part of the search space is skipped. An efficient searching for lower bounds in a partition requires problem domain knowledge. Assuming that the fitness function $F(D)$ is Lipschitzian, let $\delta(M) \equiv \max \{||x-y|| \mid x, y \in M\}$ be the diameter of M, then a lower bound is given by:

$$\beta(M) = F(y) \geq F(x) - L \parallel x - y \parallel \geq F(x) - L\delta(M), \quad \forall x, y \in M ,$$

where L is the finite Lipschitz constant. It is clear that $\beta(M) \to F(x) = \alpha$ as the diameter $\delta(M) \to 0$. Thus, we have

$$\mathrm{Lim}_{q \to \infty} (\alpha_{n_q} - \beta(M_{n_q})) = 0$$

for any decreasing sequence $\{M_{n_q}\}$ of successively refined partition elements, when $F(D)$ is Lipschitz continuous. Hence the bounding operation is *consistent*.

By keeping an ordered list of all active partitions, we can make sure that the branching operation is *complete*. That is, all active partitions are explored. Following Theorem 1, the BB-PSO algorithm converges to a global minimum in the limit.

The global convergence of BB-PSO is achieved through the branch and bound framework within which PSO operates. In the extreme case when the number of iterations within each partition is unlimited, the original search space is not partitioned. Then the BB-PSO algorithm reduces to the conventional PSO algorithm. At the other extreme, when only one iteration is allowed in each BB partition, the BB-PSO algorithm becomes traditional branch and bound with random searching for obtaining upper bounds. Because the BB part of the BB-PSO algorithm successively refines the partitions, the BB-PSO algorithm converges to a global optimum even if PSO fails to converge in each partition of the search space. Since PSO is much more effective than simple random search in reaching quality solutions, BB-PSO improves the efficiency of BB algorithms while retaining its global convergence.

Note that this theoretical convergence does not guarantee that a global optimal solution is found in practical applications. In fact, it can be shown that no algorithm can solve the problem of min $F(D)$ in a finite number of iterations. However, given an error tolerance $\varepsilon > 0$, Theorem 1 guarantees that we can find x^* such that

$$| F(x^*) - \beta(D) | \leq \varepsilon$$

in finite time. $\beta(D)$ above is the global lower bound. For many practical problems, such as many of the nonlinear benchmark test functions, $\beta(D)$ is known.

In order to obtain convergence with bound improving branching operation, we need a method to compute the lower bound $\beta(M_n)$ for each partition element M_n. Furthermore, a method must be implemented to ensure that one of the partitions with the best lower bound is selected for the next branching operation. Developing effective procedures for computing lower bounds will be carried out in future research.

5. Experiments

There are a number of nonlinear unconstrained real-valued functions that have been used as benchmark functions for testing global optimization procedures (see De Jong 1975, Davis 1991, and Reynolds & Chung 1997). We use five benchmark functions taken from Kennedy and Eberhart (2001, p.345-348) for our experiments. The five functions are unconstrained and vary in degree of complexity (see Table 1). Note that the Shaffer's f6 function has a domain of dimension two while the dimensions for the search spaces of the other functions can be arbitrarily chosen. All five functions have known global optima.

(Insert Table 1 here)

For the first experiment, we used typical parameters that had been used in published research. The choice of 0.9 for inertia weight was suggested by Shi and Eberhart (1998a). The only new parameter in Table 2 that is not used in the original PSO algorithm is the maximum number of iterations allowed in a partition. This is a parameter that we use to balance the search by PSO versus that by BB. When the maximum partition iteration is exceeded, the BB-PSO algorithm terminates the search in the current BB partition, and lets the search branch into other, potentially more promising, partitions. All the results reported in the following are the average of 20 runs with random initial particle distribution. Note that for the BB-PSO method, the distribution of particles is within the current partition, rather than the whole search space.

(Insert Table 2 here)

Table 3 gives the average number of iterations to reach an optimal solution. The optimal solution is defined as x* such as $|f(\boldsymbol{x}^*) - f(\boldsymbol{x}^t)| <$ error_tolerance, where $\boldsymbol{x}^t$ is the known global minimum. The Error tolerance used in the experiments is 0.0001. The BB-PSO algorithm terminates when an optimal solution is found or when the maximum number of iterations is reached. With the selection of proven parameters, PSO showed good performance. In general, it converged to a solution quickly, except occasional failures for the Griewank and Shaffer's functions. The failure rate is the percentage of runs that failed to reach a solution after exceeding the maximum number of iterations allowed.

By design, BB-PSO has a failure rate of zero, given a large enough maximum number of iterations. When a swarm of particles fail to find a solution in a partition, the algorithm divides the current partition into 2^d new partitions and the new partitions are merged with the active list of partitions. In Table 3, Average BB branching refers to the number of times a new partition was selected to continue the search. Since the Sphere function is easy to solve, all partition search within the first partition were successful. There was no branching needed to solve the Sphere function minimization.

(Insert Table 3 here)

Compared with PSO, BB-PSO took, in general, much more iterations to research an optimal solution. This is caused by the premature termination of the partition search that is capped at 200 iterations. The current implementation of the BB algorithm does not consider the quality of the current solutions in stopping the partition search. A modification for the algorithm could be that the partition search is terminated only if the current solution shows no improvement over several previous iterations.

The performance of PSO is sensitive to the selection of parameters. For example, Shi and Eberhart (1998b) reported that the inertia weight has a narrow range within which the PSO converges to global optimal solutions. Outside the range, the failure rate could reach to more than 50%. The advantage of BB-PSO over PSO is the guaranteed convergence to a global optimal solution. As shown in Table 3, through branching to other partitions, BB-PSO always finds a global optimal solution.

Maximum velocity also affects the performance of PSO. In our experiments, for the Schaffer's function, changing the maximum velocity from 10 to 3 increased the failure rate from 5% to 20% when the maximum number of iteration was set to 4000. In order to test the global convergence of BB-PSO, we used maximum velocity 3 to make the search more difficult in the next experiment. Compared with the results of the first experiment shown in Table 3, the average number of iterations needed to reach a solution increased significantly.

Table 4 shows the trade off between partition search (swarm within each partition) and global search (branch-and-bound). Again, the figures in Table 4 are averages over 20 runs. The number of branching and the total number of iterations are rounded to integers. The algorithm terminates when the function value of the best solution is within 0.0001 of the function value at the known optimal solution. When the number of partition iteration is limited, more branching operations are needed. When the number of partition iteration is relatively large, the need for global branching is reduced. Note that when the number of partition iteration is unlimited, BB-PSO reduces to the original PSO.

(Insert Table 4 here)

An interesting result shown in Table 4 is that balanced partition search and global search results in overall best performance in terms of total number of iterations needed to reach a global optimal solution. In this particular example, the best maximum for partition iterations is 500, which led to the smallest number of total iterations. Note that the total number of iterations includes the number of iterations used in the partitions that failed to yield an acceptable solution. The utility of the global branching is to get the PSO out of the trap once it gets stuck. Thus, BB-PSO is only beneficial when applied to difficult problems where PSO may fail to reach acceptable solutions.

Table 5 shows the experimental results with the Griewank function, for which the PSO algorithm does not always find the global optimal solution as shown in Table 3. The dimension d is set to 10. The maximum number of iteration is set to 20,000 and the error tolerance is 0.05. Pure PSO has a 35 percent failure rate in reaching an acceptable solution. With BB-PSO, similar pattern appears as in Table 4. The global optimal solution within the error tolerance is always found. The most effective strategy in terms of the least number of total iterations is to have a cap of 2000 iteration within each partition of the search space.

(Insert Table 5 here)

6. Discussion

It can be shown analytically that under certain conditions PSO may become divergent or trapped in a local minimum (Clerc & Kennedy, 2002; Van den Bergh, 2002). The BB-PSO algorithm prevents divergence by the bounding operation carried out on best performing partitions. The algorithm avoids local minimum traps by forcing the current stagnant swarm to move to a new partition, if the current partition does not produce the desired result within a given number of iterations.

Empirical experiments reveal that PSO may become stagnant for functions with complex hyper surfaces. For example, Figure 2 shows that the Griewank function has multiple local minima and those local minima are close to the global optimum both in proximity and functional values. This fact makes the Griewank function one of the most challenging non-linear functions for global optimal algorithms. Obviously, any gradient-based approach will fail to reach the global optimum unless the initial starting point falls within the attractor of the global minimum.

(Insert Figure 2 here)

Even though, by design, PSO particles may "tunnel" through the "walls" between the numerous local minima and the global minimum, they may also get trapped in local minima. With the parameter setting in Table 1 except that maximum velocity being 3 and maximum number of iterations being unlimited, experiments have shown that PSO may fail to reach the global optimal solution even after 50000 iterations. Examining the best particle solution after 10000 iterations revealed that the solution did not change with increased number of iterations. This is the situation where the BB-PSO approach will be most useful, as new partitions will be created and the most promising partitions will be selected to start a new round of PSO search. Note that although partitions that do not contain the global minimum may be selected at times, the BB algorithm ensures that those partitions will eventually be weeded out.

It should be noted that, even though the BB-PSO method ensures global convergence, exhaustive search is inherently inefficient. In the current partition scheme, each branching operation creates 2^d new partition elements. Thus the number of partitions grows exponentially with the dimension, which limits the application of BB-PSO to relatively small problems. As the dimension grows, this partition scheme quickly becomes ineffective. A practical way to overcome the "curse of dimension" is to perform one cut at each branching operation, resulting in only 2 new partitions, no matter what the dimension of the search space is. This single cut can be applied to divide the selected partition along the longest edge of the hyper-rectangle (not hyper-cube anymore).

For practical applications, a more efficient branch-and-bound scheme must be developed. A further research direction is to develop algorithms that effectively identify and eliminate partitions that do not have the potential

of containing global optimum solutions. In general, more efficient lower bounds can be found by taking advantage of the intrinsic properties of the problem at hand. Interval methods for global optimization (Hansen, 1992) may be considered to provide rigorous bounds. For continuous functions over compact domains, Lipschitz constants (Horst & Tuy, 1990) may be used to compute lower bounds in the partitions and to gauge the potential of the partitions.

7. Conclusion

Particle swarm optimization has been shown to be effective in solving a variety of complex optimization problems. With properly chosen parameters, PSO can converge to local optima. However, conventional PSO does not have guaranteed global convergence. Empirical evidence indicates that PSO algorithm may fail to reach optimal solutions for complex problems. We propose to combine the branch-and-bound framework with the particle swarm optimization algorithm. With this integrated approach, convergence to global optimal solutions is theoretically guaranteed.

We have developed and implemented the BB-PSO algorithm that combines the efficiency of PSO and effectiveness of the branch-and-bound method. When partition search is unlimited, the BB-PSO algorithm becomes the PSO algorithm. Limiting the partition search will force a stagnant swarm of particles out of the current partition, allowing the PSO to escape the trap of local minima. The BB-PSO method was tested with a set of standard benchmark nonlinear optimization problems. Experimental results show that BB-PSO achieved its goal of converging to global optimal solutions to all test problems while the PSO algorithm failed at times. With the current implementation, the BB-PSO does not scale well as the number of partitions grows exponentially with the dimension of the search space. Further research will be aimed at improving the efficiency of the BB-PSO approach by developing more efficient branching and bounding operations.

References

Angeline, P. J. (1998). Using Selection to Improve Particle Swarm Optimization. *Proc. IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, May 4-9, 1998.

Barrera, J., & Coello, C. A. C. (2009). A review of Particle Swarm Optimization Methods Used for Multimodal Optimization. *Studies in Computational Intelligence*, (24)8, 9–37.

Banks, A., Vincet, J., & Anyakoha, C. (2007). A review of particle swarm optimization. part i: background and development. *Natural Computing, 6*(4), 467–484.

Chena, R., Wub, C., Wanga, C., & Loc, S. (2010). Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB, *Expert Systems with Applications,* (37)3, 1899-1910.

Clerc, M., & Kennedy, J. (2002). The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space, *IEEE Transactions on Evolutionary Computation*, 6, 58-73.

Davis, L. (1991). *Handbook of Genetic Algorithms*. NY: van Nostrand Reinhold.

De Jong, K. (1975). *An analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan.

Eberhart, R. C., & Kennedy, J. (1995). A New Optimizer Using Particles Swarm Theory. *Proc. Sixth International Symposium on Micro Machine and Human Science.* (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, 39-43.

Eberhart, R. C., Simpson P., & Dobbins R. (1996). *Computational Intelligence PC Tools.* Academic Press

Fogel, L. J. (1994). Evolutionary Programming in Perspective: the Top-down View. *In Computational Intelligence: Imitating Life,* IEEE Press, Piscataway, NJ.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading MA: Addison-Wesley.

Hansen, E. R. (1992). *Global Optimization using Interval Analysis*. NY: Marcel Dekker.

Horst, R. (1986). A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization. *Journal of Optimization Theory and Applications,* 51, 271-291.

Horst, R., & Tuy, H. (1990). *Global Optimization: Deterministic Approaches*. Berlin: Springer-Verlag.

Hu, X., Eberhart, R. C., & Shi, Y. (2003). Engineering optimization with particle swarm. *Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2003)*, Indianapolis, Indiana, USA. 53-57.

Kennedy, J., & Eberhart, R. C. (2001). *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann Publishers.

Padhye, N., Branke, J., & Mostaghim, S. (2009). Empirical comparison of MOPSO methods: Guide selection and diversity. In *Proceedings of the 11th Congress on Evolutionary Computation*, Trondheim, Norway, 2516–2523.

Piyavskii, S. (1972). An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics,* 12, 57-67.

Reynolds, R. G., & Chung, C. J. (1997). A Test Bed for Solving Optimization Problems Using Cultural Algorithms. *Proceedings of Fifth Annual Conference on Evolutionary Programming*, San Diego, California.

Shi, Y. H. & Eberhart, R. C. (1998a). A Modified Particle Swarm Optimizer. *IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, May 4-9, 1998.

Shi, Y. H., & Eberhart, R. C. (1998b). Parameter selection in particle swarm optimization. *Proceedings of the 1998 Annual Conference on Evolutionary Programming,* San Diego, CA.

Van den Bergh, F. (2002). *An analysis of particle swarm optimizers*. Ph.D. Dissertation, Department of Computer Science, University of Pretoria, South Africa.

Van den Bergh, F.,& Engelbrecht, A. P. (2002). A new locally convergent particle swarm optimizer. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics 2002*, 96-101.

Venter, G., & Haftka, R. T. (2010). Constrained particle swarm optimization using a bi-objective formulation. *Structural and Multidisciplinary Optimization*, (40)1-6, 65-76.

Table 1. Benchmark functions used in the experiments

Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$
Rosenbrock	$f_2(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$
Rastrigin	$f_3(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$
Griewank	$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Shaffer's f6	$f_5(x) = 0.5 - \frac{(\sin\sqrt{x_1^2 + x_2^2})^2 - 0.5}{(1.0 + 0.001(x_1^2 + x_2^2)^2}$

Table 2. Experiment parameters

Swarm population size	20
Dimension	2
Max velocity	10
Max domain range	±100
Max iteration	2000
Max partition iteration	200
Error tolerance	0.0001
Inertia weight	0.9

Table 3. Average number of iterations to reach a global optimal solution

Function	PSO		BB-PSO	
	Average iter #	Failure rate	Average BB branching	Average total iter #
Sphere	68	0%	0	65
Rosenbrock	253	0%	2	591
Rastrigin	168	0%	1	203
Griewank	305*	10%	5	931
Shaffer's f6	238*	5%	4	534

* The average number of iterations is computed excluding those runs that failed to find an optimal solution within the maximum of 4000 iterations.

Table 4. Average number of iterations for Schaffer's function (d=2)

Max partition iteration	**# of Branching**	**Total # of iterations**
200	14	2989
500	3	1690
1000	2	2326
2000	1	3388

Table 5. Average number of iterations for Griewank function (d=10)

Max partition iteration	**# of Branching**	**Total # of iterations**
1000	15	15184
2000	5	10952
4000	3	12186
8000	2	16337

ISSN 1913-8989 E-ISSN 1913-8997

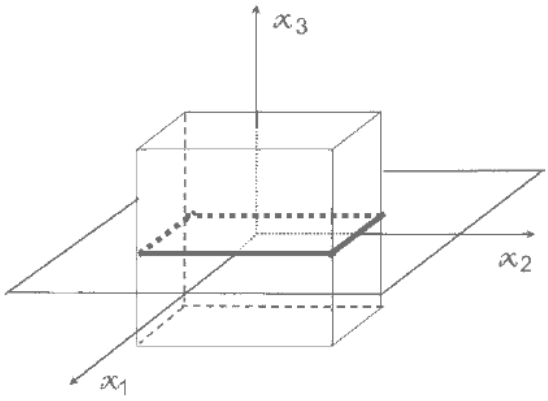

Figure 1. Dividing a (hyper) cube with a (hyper) plane

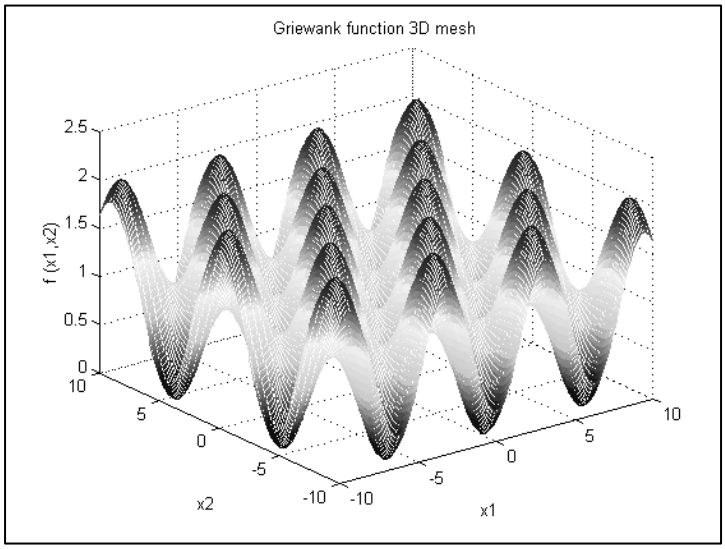

Figure 2. 3D mesh of the Griewank function