Barebones Particle Swarm for Integer Programming Problems

Mahamed G. H. Omran, Andries Engelbrecht and Ayed Salman

Abstract—**The performance of two recent variants of Particle Swarm Optimization (PSO) when applied to Integer Programming problems is investigated. The two PSO variants, namely, barebones Particle Swarm (BB) and the exploiting barebones Particle Swarm (BBExp) are compared with the standard PSO and standard Differential Evolution (DE) on several Integer Programming test problems. The results show that the BBExp seems to be an efficient alternative for solving Integer Programming problems.**

I. INTRODUCTION

MANY real-world applications (e.g. production scheduling, resource allocation, VLSI circuit design, etc.) require the variables to be optimized to be integers. These problems are called Integer Programming problems. Optimization methods developed for real search spaces can be used to solve Integer Programming problems by rounding off the real optimum values to the nearest integers [1].

The unconstrained Integer Programming problem can be defined as

$$\min f(\boldsymbol{x}), \boldsymbol{x} \in \boldsymbol{S} \subseteq \boldsymbol{Z}^{N_d} \tag{1}$$

where $\boldsymbol{Z}^{N_d}$ is an N_d–dimensional discrete space of integers, and $\boldsymbol{S}$ represents a feasible region that is not necessarily a bounded set. Integer Programming problems encompass both maximization and minimization problems. Any maximization problem can be converted into a minimization problem and *vice versa*. The problems tackled in this paper are minimization problems. Therefore, the remainder of the discussion focuses on minimization problems.

The Branch and Bound method [2] is one common deterministic approach to tackle the Integer Programming problems. Evolutionary algorithms (EAs) [3] are general-purpose stochastic search methods simulating natural selection and evolution in the biological world. EAs have been used successfully to solve Integer Programming problems [4]. On the other hand, Particle Swarm Optimization (PSO) [5] is a population-based stochastic optimization algorithm modeled after the simulation of the social behavior of bird flocks, where a swarm of individuals (called *particles*) fly through the search space. PSO has also been used to solve Integer Programming problems with promising results [1,6,7].

Kennedy [8] proposed a new PSO approach where the standard PSO velocity equation is removed and replaced with samples from a normal distribution. This approach, known as barebones Particle Swarm (BB), requires no parameter tuning. Kennedy [8] proposed a variation of the BB PSO where approximately half of the time velocity is based on samples from a normal distribution, and for half of the time velocity is derived from the particle's personal best position. This version is called exploiting barebones Particle Swarm (BBExp).

This paper investigates the performance of BB and BBExp methods on Integer Programming problems. The two variants are compared to both PSO (as proposed in [1]) and standard Differential Evolution (DE) [9].

The reminder of the paper is organized as follows: Section II provides an overview of PSO, BB and BBExp. An overview of DE is given in Section III. Benchmark functions to measure the performance of the different approaches are provided in Section IV. Results of the experiments are presented in Section V. Finally, Section VI concludes the paper.

II. PARTICLE SWARM OPTIMIZATION AND ITS VARIANTS

In PSO, each particle in the swarm is represented by the following characteristics:

$\boldsymbol{x}_i$: The *current position* of the particle;
$\boldsymbol{v}_i$: The *current velocity* of the particle;
$\boldsymbol{y}_i$: The *personal best position* of the particle.

The personal best position of particle i is the best position (i.e. one resulting in the best fitness value) visited by particle i so far. Let f denote the objective function. Then the personal best of a particle at time step t is updated as

$$\boldsymbol{y}_i(t+1) = \begin{cases} \boldsymbol{y}_i(t) & \text{if } f(\boldsymbol{x}_i(t+1)) \geq f(\boldsymbol{y}_i(t)) \\ \boldsymbol{x}_i(t+1) & \text{if } f(\boldsymbol{x}_i(t+1)) < f(\boldsymbol{y}_i(t)) \end{cases} \tag{2}$$

Manuscript received November 3, 2006.

M. G. Omran is with the Department of Computer Science, Gulf University for Science and Technology, Kuwait (phone: 886644 (Ext. 5542); e-mail: mjomran@gmail.com).

A. Engelbrecht is with the Department of Computer Science, University of Pretoria, Pretoria, South Africa.

A. Salman with the Department of Computer Engineering, Kuwait University, Kuwait.

1-4244-0708-7/07/$20.00 ©2007 IEEE

If the position of the global best particle is denoted by the vector $\hat{\boldsymbol{y}}$, then

$$\hat{\boldsymbol{y}}(t) \in \{\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_s\} = \min\{f(\boldsymbol{y}_0(t)), f(\boldsymbol{y}_1(t)), \ldots, f(\boldsymbol{y}_s(t))\} \quad (3)$$

where s denotes the size of the swarm.

For each iteration of a PSO algorithm, the velocity $\boldsymbol{v}_i$ update step is specified for each dimension $j \in 1,\ldots, N_d$, where N_d is the dimension of the problem. Hence, $v_{i,j}$ represents the j^{th} element of the velocity vector of the i^{th} particle. The velocity of particle i is updated as

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t)) \quad (4)$$

where w is the inertia weight, c_1 and c_2 are the acceleration constants and $r_{1,j}, r_{2,j} \sim U(0,1)$. Equation (4) consists of three components, namely

- The *inertia weight* term, w, which serves as a memory of previous velocities. The inertia weight controls the impact of the previous velocity: a large inertia weight favors exploration, while a small inertia weight favors exploitation.
- The cognitive component, $\boldsymbol{y}_i - \boldsymbol{x}_i$, which represents the particle's own experience as to where the best solution is.
- The social component, $\hat{\boldsymbol{y}} - \boldsymbol{x}_i$, which represents the belief of the entire swarm as to where the best solution is.

The position of particle i, $\boldsymbol{x}_i$, is then updated using

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \quad (5)$$

The reader is referred to [10,11] for a study of the relationship between the inertia weight and acceleration constants, in order to select values for these control parameters which will ensure convergent behavior. Velocity updates can also be clamped through a user defined maximum velocity, $V_{\max}$, which would prevent them from exploding, thereby reducing the chances that particles will leave the boundaries of the search space.

The PSO algorithm performs the update equations above, repeatedly, until a specified number of iterations have been exceeded, or velocity updates are close to zero. The quality of particles is measured using a fitness function which reflects the optimality of a particular solution.

A. Barebones Particle Swarm approaches

BB PSO replaces equations (4) and (5) with the following equation,

$$x_{i,j}(t+1) = N\left(\frac{y_{i,j}(t) + \hat{y}_j(t)}{2}, \left|y_{i,j}(t) - \hat{y}_j(t)\right|\right) \quad (6)$$

Equation (6) is based on theoretical studies where formal proofs have shown that each particle converges to a weighted average of its personal best and neighborhood best positions [10,11]. Therefore, the mean of

$$\frac{y_{i,j}(t) + \hat{y}_j(t)}{2}$$

is used for the normal distribution. The deviation of

$$y_{i,j}(t) - \hat{y}_j(t)$$

allows particles whose personal best position are far away from the global best position to make large step sizes towards the global best position. This may cause personal best positions to move closer to the global best position. When a personal best position is close to the global best position, step sizes are small to limit exploration in favor of exploitation.

The BBExp, on the other hand, replaces equations (4) and (5) with

$$x_{i,j}(t+1) \begin{cases} N\left(\frac{y_{i,j}(t) + \hat{y}_j(t)}{2}, \left|y_{i,j}(t) - \hat{y}_j(t)\right|\right) & \text{if } U(0,1) > 0.5 \\ y_{i,j}(t) & \text{otherwise} \end{cases} \quad (7)$$

The fact that position updates are set equal to the personal best position for 50% of the time causes the BBExp to exploit personal best positions more than BB PSO, thereby limiting exploration.

According to [8], BBExp generally outperformed other variants of PSO when applied to a set of benchmark functions.

III. DIFFERENTIAL EVOLUTION

Differential evolution does not make use of a mutation operator that depends on some probability distribution function, but introduces a new arithmetic operator which depends on the differences between randomly selected pairs of individuals.

For each parent, $\boldsymbol{x}_i(t)$, of generation t, an offspring, $\boldsymbol{x}'_i(t)$, is created in the following way: Randomly select three individuals from the current population, namely $\boldsymbol{x}_{i_1}(t)$,

$\boldsymbol{x}_{i_2}(t)$ and $\boldsymbol{x}_{i_3}(t)$, with $i_1 \neq i_2 \neq i_3 \neq i$ and i_1, i_2, $i_3 \sim U(1,\ldots, s)$, where s is the population size. Select a random number $r \sim U(1,\ldots, N_d)$, where N_d is the number of genes (parameters) of a single chromosome. Then, for all parameters $j = 1,\ldots, N_d$, if $U(0, 1) < P_r$, or if $j = r$, let

$$x'_{i,j}(t) = x_{i_3,j}(t) + F(x_{i_1,j}(t) - x_{i_2,j}(t)) \quad (8)$$

otherwise, let

$$x'_{i,j}(t) = x_{i,j}(t) \quad (9)$$

In the above, P_r is the probability of reproduction (with $P_r \in [0, 1]$), F is a scaling factor with $F \in (0, 8)$, and $x'_{i,j}(t)$ and $x_{i,j}(t)$ indicate respectively the j-th parameter of the offspring and the parent.

Thus, each offspring consists of a linear combination of three randomly chosen individuals when $U(0,1) < P_r$; otherwise the offspring inherits directly from the parent. Even when $P_r = 0$, at least one of the parameters of the offspring will differ from the parent (forced by the condition $j = r$).

The mutation process above requires that the population consists of more than three individuals.

After completion of the mutation process, the next step is to select the new generation. For each parent of the current population, the parent is replaced with its offspring if the fitness of the offspring is better, otherwise the parent is carried over to the next generation.

Price and Storn [12] proposed ten different strategies for DE based on the individual being perturbed (i.e. $\boldsymbol{x}_{i_3,j}(t)$), number of individuals used in the mutation process and the type of crossover used. The strategy shown in this section is known as DE/rand/1. This strategy is considered to be the most widely used strategy.

IV. Benchmark Functions

Six commonly used Integer programming benchmark problems [1] were chosen to investigate the performance of the PSO variants. Elements of the particle position vectors are rounded to the nearest integer, after the application of the position update equation.

Test Problem 1:

$$\boldsymbol{F}_1(\boldsymbol{x}) = \sum_{i=1}^{N_d} |x_i|$$

where $\boldsymbol{x}^* = \boldsymbol{0}$ and $f(\boldsymbol{x}^*) = 0$. This problem was considered for dimensions 5, 15 and 30.

Test Problem 2:

$$F_2(x) = \left(9x_1^2 + 2x_2^2 - 11\right)^2 + \left(3x_1 + 4x_2^2 - 7\right)^2$$

where $\boldsymbol{x}^* = (\mathbf{1},\mathbf{1})^{\mathrm{T}}$ and $F_2(\boldsymbol{x}^*) = 0$.

Test Problem 3:

$$F_3(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

where $\boldsymbol{x}^* = \mathbf{0}$ and $F_3(\boldsymbol{x}^*) = 0$.

Test Problem 4:

$$F_4(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2$$

where $\boldsymbol{x}^* = \mathbf{0}$ and $F_4(\boldsymbol{x}^*) = 0$.

Test Problem 5:

$$F_5(x) = -380384 - 13808x_1 - 23292x_2 + 12308x_1^2 + 20364x_2^2 + 18225x_1x_2$$

where $\boldsymbol{x}^* = (\mathbf{0},\mathbf{1})^{\mathrm{T}}$ and $F_5(\boldsymbol{x}^*) = -3833.12$.

Test Problem 6:

$$\boldsymbol{F}_6(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$$

where $\boldsymbol{x}^* = \mathbf{0}$ and $f(\boldsymbol{x}^*) = 0$. This problem was considered for dimension 5 as in [1].

For all the above problems, $x_i \in [-100,100] \subseteq \boldsymbol{Z}$.

V. Experimental Results

In this section, BB and BBExp are compared with standard *gbest* PSO and DE/rand/1. For PSO, w =0.72, $c_1 = c_2 = 1.49$ (these values were suggested by [13]). For DE, $F = 0.5$, $P_r = 0.9$ (these values were suggested by [12]). For BB and BBExp, no control parameter tuning is needed (except for the swarm size s). For all the algorithms used in this section, $s = 50$.

The results reported in this section are averages and standard deviations over 30 simulations. Each simulation was allowed to run for 50 000 evaluations of the objective function.

Table I summarizes the results of the experiments. For all

the test problems, except for F_1 with $N_d = 15$ and F_1 with $N_d = 30$, all the algorithms found the global optimum solutions. For larger dimensional problems (F_1 with $N_d = 30$), the BBExp performed best, giving significantly better results than the other algorithms. What is interesting to note is that the BB PSO performed worse than the other algorithms for F_1 with $N_d = 15$ and $N_d = 30$. This may be due to larger step sizes caused by the deviation, $\left|y_{i,j}(t) - \hat{y}_j(t)\right|$, of the normal distribution.

Examining the results, BBExp significantly outperformed the other approaches when applied to the 30-dimensional F_1. BB performed worse than the other approaches when applied to F_1. Examining Figure 1 it can be shown that BBExp converges faster than the other approaches when applied to the 30-dimensional F_1. Furthermore, Figure 1 shows that BB suffers from premature convergence. The expected reason for the premature convergence is that BB lost its diversity faster than the other approaches as shown in Fig. 2. For all the other functions, all the approaches performed equally well and exhibit similar convergence characteristics. Although the algorithms achieved the same accuracy for the lower dimensional problems, BBExp have shown the following advantages:

- Better accuracy for the higher dimensional problems.
- Faster convergence.
- Requires no parameter tuning, whereas the performance of PSO and DE is dependent on values selected for their control parameters.

VI. CONCLUSION

This paper evaluated the performance of two version of the barebones PSO in solving Integer Programming problems. In comparison with a *gbest* PSO and DE/rand/1, the algorithms showed the same performance in terms of accuracy for the lower dimensional problems. For higher dimensional problems, BBExp showed to be significantly better than the other algorithms. The results also showed that BBExp converges faster to good solutions.

REFERENCES

[1] E. Laskari, K. Parsopoulos and M. Vrahatis. Particle Swarm Optimization for Integer Programming. In: Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2, pp. 1582-1587, 2002.

[2] R. Horst and H. Tuy. Global Optimization, Deterministic Approaches, Springer, 1996.

[3] A. Engelbrecht. Computational Intelligence: An Introduction. John Wiley and Sons, 2002.

[4] G. Rüdolph. An Evolutionary Algorithm for Integer Programming. Y. Davidoe, H. Schwefel and R. Männer (eds.), Parallel Problem Solving from Nature, vol. 3, pp. 139-148, Springer, 1994.

[5] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks*, Perth, Australia, vol. 4, pp. 1942-1948, 1995.

[6] T. Tsukada, T. Tamura, S. Kitagawa and Y. Fukuyama. Optimal operational planning for cogeneration system using particle swarm optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, Indianapolis, Indiana, USA. pp. 138-143, 2003.

[7] K. Parsopoulos and M. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, vol. 1, no. 2-3, pp. 235-306, 2002.

[8] J. Kennedy. Bare bones particle swarm, *IEEE Swarm Intelligence Symposium*, pp. 80-87, 2003.

[9] R. Storn and K. Price. Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, 1995.

[10] F. Van den Bergh. *An analysis of particle swarm optimizers, Ph.D. dissertation*. Department of Computer Science, University of Pretoria, 2002.

[11] F. Van den Bergh and A. P. Engelbrecht. A Study of Particle Trajectories. *Information Sciences*, vol. 176, no. 8, pp. 937-971, 2006.

[12] K. Price and R. Storn. DE Web site, http://www.ICSI.Berkeley.edu/~storn/code.html (visited 8 Aug 2006), 2006.

[13] M. Clerc and J. Kennedy. The particle swarm: Explosion, Stability and Convergence in a multi-dimentional complex space. *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58-73, 2002.

TABLE I
MEAN, STANDARD DEVIATION (SD) AND 95% CONFIDENCE INTERVAL OF THE TEST PROBLEMS

Function	Method	Mean (SD)	95% Confidence Interval (z-distribution)
F_1 ($N_d = 5$)	PSO	0(0)	[0,0]
	DE	0(0)	[0,0]
	BB	0(0)	[0,0]
	BBExp	0(0)	[0,0]
F_1 ($N_d = 15$)	PSO	0(0)	[0,0]
	DE	0(0)	[0,0]
	BB	0.433333(0.897634)	[0.201565,0.665101]
	BBExp	0(0)	[0,0]
F_1 ($N_d = 30$)	PSO	4.066667(18.339581)	[0.718333,7.415001]
	DE	1.5 (1.978331)	[1.138808,1.861192]
	BB	10.600000(6.234498)	[9.461742,11.738258]
	BBExp	0.366667(0.556053)	[0.265146,0.468188]
F_2	PSO	0(0)	[0,0]
	DE	0(0)	[0,0]
	BB	0(0)	[0,0]
	BBExp	0(0)	[0,0]
F_3	PSO	0(0)	[0,0]
	DE	0(0)	[0,0]
	BB	0(0)	[0,0]
	BBExp	0(0)	[0,0]
F_4	PSO	-6(0)	[-6,-6]
	DE	-6(0)	[-6,-6]
	BB	-6(0)	[-6,-6]
	BBExp	-6(0)	[-6,-6]
F_5	PSO	-3833.12(0)	[-3833.12,-3833.12]
	DE	-3833.12(0)	[-3833.12,-3833.12]
	BB	-3833.12(0)	[-3833.12,-3833.12]
	BBExp	-3833.12(0)	[-3833.12,-3833.12]
F_6 ($N_d = 5$)	PSO	0(0)	[0,0]
	DE	0(0)	[0,0]
	BB	0(0)	[0,0]
	BBExp	0(0)	[0,0]

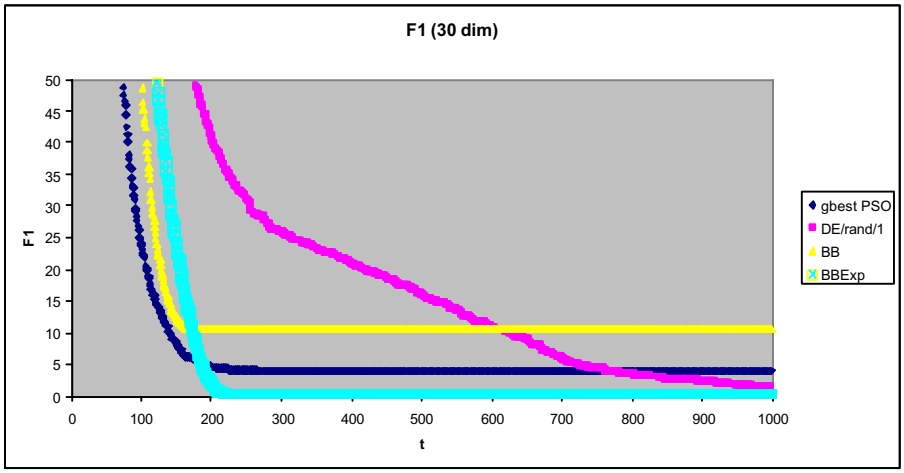

Fig. 1: Comparison between PSO, DE, BB and BBExp for the 30-dimensional F_1 benchmark problem. The vertical axis represents the function value and the horizontal axis represents the number of generations.

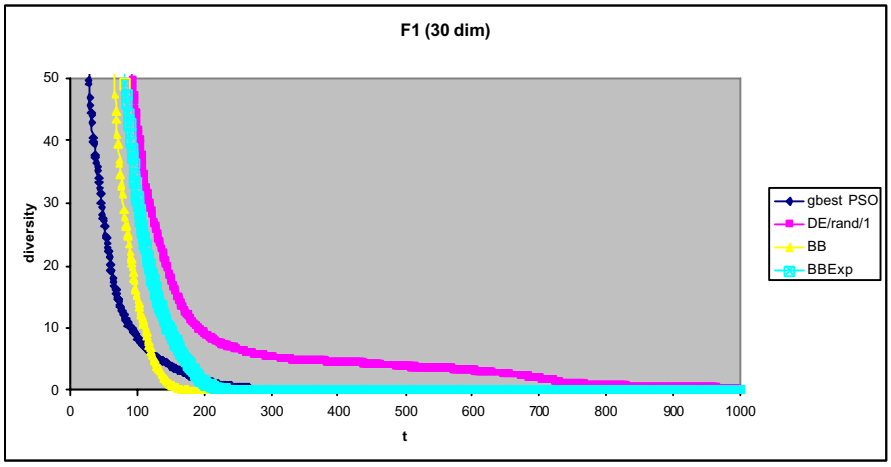

Fig. 2: Comparison between PSO, DE/rand/1, BB and BBExp for the 30-dimensional F_1 benchmark problem. The vertical axis represents the diversity and the horizontal axis represents the number of generations.