UNIFIED PARTICLE SWARM OPTIMIZATION FOR TACKLING OPERATIONS RESEARCH PROBLEMS

K.E. Parsopoulos and M.N. Vrahatis

Computational Intelligence Laboratory (CI Lab), Department of Mathematics,
and
University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR-26110 Patras, Greece
{kostasp,vrahatis}@math.upatras.gr

ABSTRACT

We investigate the performance of the recently proposed Unified Particle Swarm Optimization algorithm on two categories of operations research problems, namely minimax and integer programming problems. Different variants of the algorithm are employed and compared with established variants of the Particle Swarm Optimization algorithm. Statistical hypothesis testing is performed to justify the significance of the results. Conclusions regarding the ability of the Unified Particle Swarm Optimization method to tackle operations research problems as well as on the performance of each variant are derived and discussed.

1. INTRODUCTION

Two of the most interesting categories of problems in operations research are minimax and integer programming problems [1]. Such problems are encountered in numerous engineering and scientific applications, including optimal control, engineering design, game theory, molecular biology, high energy physics, capital budgeting and portfolio analysis [2, 3].

In general, the minimax problem can be defined as

$$\min_x F(x), \tag{1}$$

where,

$$F(x) = \max_{i=1,\ldots,m} f_i(x), \tag{2}$$

with $f_i(x) : S \subset \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$. Moreover, a nonlinear programming problem with inequality constraints of the form

$$\begin{aligned} &\min F(x), \\ &\text{subject to} \quad g_i(x) \geqslant 0, \quad i = 2, \ldots, m, \end{aligned} \tag{3}$$

can be transformed into the minimax problem

$$\min_x \max_{i=1,\ldots,m} f_i(x),$$

where,

$$\begin{aligned} f_1(x) &= F(x), \\ f_i(x) &= F(x) - \alpha_i g_i(x), \\ \alpha_i &> 0, \end{aligned} \tag{4}$$

for $i = 2, \ldots, m$. It has been proved that for sufficiently large α_i, the optimum point of the minimax problem coincides with the optimum point of the nonlinear programming problem [4]. Minimax problems have proved to be difficult to be tackled through traditional gradient–based algorithms, since, at points where $f_j(x) = F(x)$ for two or more values of $j \in \{1, \ldots, m\}$, the first partial derivatives of $F(x)$ are discontinuous, even if all the functions $f_i(x)$, $i = 1, \ldots, m$, have continuous first partial derivatives.

The unconstrained integer programming problem is defined as

$$\min_x f(x), \quad x \in S \subseteq \mathbb{Z}^n, \tag{5}$$

where $\mathbb{Z}$ is the set of integers, and S is a not necessarily bounded set, which is considered as the feasible region. In this first investigation we focus to all–integer programming problems, where all variables are integers. Mixed–integer programming problems, where some of the variables are real, will be considered in future works.

Particle Swarm Optimization (PSO) has proved to be very efficient algorithm for addressing minimax and integer programming problems [5, 6]. The performance of population–based algorithms is heavily dependent on the trade–off between their exploration and exploitation capabilities. To this end, Unified Particle Swarm Optimization (UPSO) was recently introduced as a unified PSO scheme that combines the exploration and exploitation properties of different PSO variants [7]. Preliminary results on static as well as dynamic optimization problems indicate the superiority of UPSO against the standard PSO variants [7, 8].

We investigate the performance of UPSO on minimax and integer programming problems and compare it with the

performance of the local and global PSO variant on well–known benchmark functions. Statistical hypothesis testing is conducted to justify the significance of the results. The rest of the paper is organized as follows. PSO and UPSO are briefly described in Section 2 and experimental results are reported and discussed in Section 3. The paper closes with conclusions in Section 4.

2. UNIFIED PARTICLE SWARM OPTIMIZATION

PSO is the most common swarm intelligence algorithm for numerical optimization tasks. It was introduced in 1995 by Eberhart and Kennedy [9, 10], drawing inspiration from the emergent behavior in socially organized colonies [11]. PSO is a population–based algorithm, i.e., it employs a population, called a *swarm*, of search points,

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top, \quad i = 1, \ldots, N,$$

called *particles*, to probe the search space, S. Each particle moves in S with an adaptable velocity,

$$v_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top,$$

and stores the best position,

$$p_i = (p_{i1}, p_{i2}, \ldots, p_{in})^\top \in S,$$

it has ever visited in the search space. Also, each particle is considered to have a neighborhood that consists of a number of other particles and its movement is influenced by their experience (i.e., best positions).

The neighborhoods can be defined in different ways. Different topologies have been proposed and applied with promising results [12, 13]. The most common neighborhood topology is the ring topology, where the immediate neighbors of the particle x_i are the particles x_{i-1}, x_{i+1}, and x_1 is considered to be the particle that follows immediately after x_N. Thus, a neighborhood of radius M of x_i consists of the particles $x_{i-M}, \ldots, x_i, \ldots, x_{i+M}$. The established ring topology is the scheme that we adopted in the current study. There are two main variants of PSO with respect to the number of particles that comprise the neighborhood of a particle. In the *global* variant, the whole swarm is considered as the neighborhood of each particle, while, in the *local* variant, smaller neighborhoods are used.

Let g_i be the index of the best particle in the neighborhood of x_i, i.e., the index of the particle that attained the best position among all the particles of the neighborhood. Then, the swarm is updated according to the equations [14],

$$v_i^{(k+1)} = \chi \left[v_i^{(k)} + \varphi_1 \left(p_i^{(k)} - x_i^{(k)} \right) + \varphi_2 \left(p_{g_i}^{(k)} - x_i^{(k)} \right) \right] \quad (6)$$

$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)}, \quad (7)$$

where $i = 1, \ldots, N$; k is the iteration counter; χ is a parameter called *constriction coefficient* that controls the velocity's magnitude; $\varphi_1 = c_1 r_1$ and $\varphi_2 = c_2 r_2$, where c_1 and c_2 are positive acceleration parameters, called *cognitive* and *social* parameter, respectively, and r_1, r_2 are random vectors that consist of random values uniformly distributed in $[0, 1]$. All vector operations in Eqs. (6) and (7) are performed componentwise. A stability analysis of PSO, as well as recommendations regarding the selection of its parameters are provided in [14, 15].

The dependence of an algorithm's performance on the balance between its exploration and exploitation ability, i.e., its ability to perform global search of the search space and converge faster to the most promising regions, respectively, triggered the development of UPSO. More specifically, in the global variant of PSO, all particles are attracted by the same best position, converging faster towards specific points. Thus, it has better exploitation abilities in contrast to the local variant, where the information of the best position of each neighborhood is communicated slowly to the other particles of the swarm through their neighbors in the ring topology, thereby promoting exploration.

UPSO harnesses the two PSO variants in a unified scheme that combines their exploration and exploitation capabilities [7]. Let $\mathcal{G}_i^{(k+1)}$ denote the velocity update of the particle x_i in the global PSO variant and let $\mathcal{L}_i^{(k+1)}$ denote the corresponding velocity update for the local variant. Then, according to Eq. (6),

$$\mathcal{G}_i^{(k+1)} = \chi \left[v_i^{(k)} + \varphi_1 \left(p_i^{(k)} - x_i^{(k)} \right) + \varphi_2 \left(p_g^{(k)} - x_i^{(k)} \right) \right], \quad (8)$$

$$\mathcal{L}_i^{(k+1)} = \chi \left[v_i^{(k)} + \varphi_1' \left(p_i^{(k)} - x_i^{(k)} \right) + \varphi_2' \left(p_{g_i}^{(k)} - x_i^{(k)} \right) \right], \quad (9)$$

where k denotes the iteration number; g is the index of the best particle of the whole swarm (global variant); and g_i is the index of the best particle in the neighborhood of x_i (local variant). These two search directions are combined in a single equation, resulting in the main UPSO scheme [7],

$$\mathcal{U}_i^{(k+1)} = u\, \mathcal{G}_i^{(k+1)} + (1-u)\, \mathcal{L}_i^{(k+1)}, \quad (10)$$

$$x_i^{(k+1)} = x_i^{(k)} + \mathcal{U}_i^{(k+1)}, \quad (11)$$

where $u \in [0, 1]$ is called the *unification factor* and it determines the influence of the global and local search direction in Eq. (10). The standard local and global PSO variant is obtained for $u = 0$ and $u = 1$, respectively, while, for all intermediate values $u \in (0, 1)$, we obtain composite variants of PSO that combine the exploration and exploitation characteristics of the global and local variant.

UPSO can be further enhanced by incorporating a stochastic parameter in Eq. (10). This parameter imitates mutation in evolutionary algorithms, although, it is directed towards a direction that is consistent with the PSO dynamic. Thus, Eq. (10) can be written either as

$$\mathcal{U}_i^{(k+1)} = r_3\, u\, \mathcal{G}_i^{(k+1)} + (1-u)\, \mathcal{L}_i^{(k+1)}, \quad (12)$$

which is mostly based on the local variant or, alternatively,

$$\mathcal{U}_i^{(k+1)} = u\, \mathcal{G}_i^{(k+1)} + r_3\, (1-u)\, \mathcal{L}_i^{(k+1)}, \quad (13)$$

which is mostly based on the global variant, where $r_3 \sim \mathcal{N}(M, \Sigma)$ is a normally distributed parameter with mean vector M and variance matrix Σ. Based on the analysis of Matyas [16] for stochastic optimization algorithms, convergence in probability was proved for the schemes of Eqs. (12) and (13) [7].

3. RESULTS

UPSO was applied on 10 minimax and 7 integer programming benchmark problems on which PSO was recently tested [5, 6]. These problems are defined as follows [17, 18, 19, 20, 21, 22].

MINIMAX PROBLEMS

Test Problem 1 [17]. This is a 2–dimensional problem and it consists of 3 functions,

$$\begin{aligned} & \min_x F_1(x), \\ F_1(x) &= \max\{f_i(x)\}, \quad i = 1, 2, 3, \\ f_1(x) &= x_1^2 + x_2^4, \\ f_2(x) &= (2 - x_1)^2 + (2 - x_2)^2, \\ f_3(x) &= 2\exp(-x_1 + x_2). \end{aligned} \tag{14}$$

Test Problem 2 [17]. This is a 2–dimensional problem and it consists of 3 functions,

$$\begin{aligned} & \min_x F_2(x), \\ F_2(x) &= \max\{f_i(x)\}, \; i = 1, 2, 3, \\ f_1(x) &= x_1^4 + x_2^2, \\ f_2(x) &= (2 - x_1)^2 + (2 - x_2)^2, \\ f_3(x) &= 2\exp(-x_1 + x_2). \end{aligned} \tag{15}$$

Test Problem 3 [17]. This is a 4–dimensional nonlinear programming problem and it is transformed to an equivalent minimax problem according to Eq. (4),

$$\begin{aligned} F_3(x) &= x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4, \\ g_2(x) &= -x_1^2 - x_2^2 - x_3^3 - x_4^2 - x_1 + x_2 - x_3 + x_4 + 8 \\ g_3(x) &= -x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2 + x_1 + x_4 + 10, \\ g_4(x) &= -x_1^2 - x_2^2 - x_3^2 - 2x_1 + x_2 + x_4 + 5. \end{aligned} \tag{16}$$

Test Problem 4 [17]. This is a 5–dimensional nonlinear programming problem and it is transformed to an equivalent minimax problem according to Eq. (4),

$$\begin{aligned} F_4(x) &= (x_1 - 10)^2 + 5(x_2 - 12)^2 + 3(x_4 - 11)^2 + x_3^4 + \\ &\quad +10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7, \\ g_2(x) &= -2x_1^2 - 3x_3^4 - x_3 - 4x_4^2 - 5x_5 + 127, \\ g_3(x) &= -7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 + 282, \\ g_4(x) &= -23x_1 - x_2^2 - 6x_6^2 + 8x_7 + 196, \\ g_5(x) &= -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7. \end{aligned} \tag{17}$$

Test Problem 5 [18]. This is a 2–dimensional problem and it consists of 2 functions,

$$\begin{aligned} & \min_x F_5(x), \\ F_5(x) &= \max\{f_i(x)\}, \; i = 1, 2, \\ f_1(x) &= |x_1 + 2x_2 - 7|, \\ f_2(x) &= |2x_1 + x_2 - 5|. \end{aligned} \tag{18}$$

Test Problem 6 [18]. This is a 10–dimensional problem and it consists of 10 functions,

$$\begin{aligned} & \min_x F_6(x), \\ F_6(x) &= \max\{f_i(x)\}, \\ f_i(x) &= |x_i|, \quad i = 1, \ldots, 10. \end{aligned} \tag{19}$$

Test Problem 7 [19]. This is a 2–dimensional problem and it consists of 2 functions,

$$\begin{aligned} & \min_x F_7(x), \\ F_7(x) &= \max\{f_i(x)\}, \\ f_1(x) &= (x_1 - b\cos b)^2 + 0.005\left(x_1^2 + x_2^2\right), \\ f_2(x) &= (x_2 - b\sin b)^2 + 0.005\left(x_1^2 + x_2^2\right), \end{aligned} \tag{20}$$

were $b = \sqrt{x_1^2 + x_2^2}$.

Test Problem 8 [19]. This is a 4–dimensional problem and it consists of 4 functions,

$$\begin{aligned} & \min_x F_8(x), \\ F_8(x) &= \max\{f_i(x)\}, \quad i = 1, \ldots, 4, \\ f_1(x) &= \Big(x_1 - (x_4 + 1)^4\Big)^2 + \Big(x_2 - \Big(x_1 - (x_4 + 1)^4\Big)^4\Big)^2 + \\ &\quad 2x_3^2 + x_4^2 - 5\Big(x_1 - (x_4 + 1)^4\Big) - 5\Big(x_2 - \Big(x_1 - \\ &\quad (x_4 + 1)^4\Big)^4\Big) - 21x_3 + 7x_4, \\ f_2(x) &= f_1(x) + 10\Bigg[\Big(x_1 - (x_4 + 1)^4\Big)^2 + \Big(x_2 - \Big(x_1 - \\ &\quad (x_4 + 1)^4\Big)^4\Big)^2 + x_3^2 + x_4^2 + \Big(x_1 - (x_4 + 1)^4\Big) - \\ &\quad \Big(x_2 - \Big(x_1 - (x_4 + 1)^4\Big)^4\Big) + x_3 - x_4 - 8\Bigg], \\ f_3(x) &= f_1(x) + 10\Bigg[\Big(x_1 - (x_4 + 1)^4\Big)^2 + 2\Big(x_2 - \Big(x_1 - \\ &\quad (x_4 + 1)^4\Big)^4\Big)^2 + x_3^2 + 2x_4^2 - \Big(x_1 - (x_4 + 1)^4\Big) - \\ &\quad x_4 - 10\Bigg], \end{aligned} \tag{21}$$

$$f_4(x) = f_1(x) + 10\Big[\Big(x_1 - (x_4+1)^4\Big)^2 + \Big(x_2 - \Big(x_1 - (x_4+1)^4\Big)^4\Big)^2 + x_3^2 + 2\Big(x_1 - (x_4+1)^4\Big) - \Big(x_2 - \Big(x_1 - (x_4+1)^4\Big)^4\Big) - x_4 - 5\Big].$$

Test Problem 9 [19]. This is a 7–dimensional problem and it consists of 5 functions,

$$\min_x F_9(x),$$

$$\begin{aligned}
F_9(x) &= \max\{f_i(x)\}, \quad i = 1,\ldots,5,\\
f_1(x) &= (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7\\
f_2(x) &= f_1(x) + 10(2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127),\\
f_3(x) &= f_1(x) + 10(7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282),\\
f_4(x) &= f_1(x) + 10(23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196),\\
f_5(x) &= f_1(x) + 10(4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7).
\end{aligned} \tag{22}$$

Test Problem 10 [19]. This is a 4–dimensional problem and it consists of 21 functions,

$$\min_x F_{10}(x),$$

$$\begin{aligned}
F_{10}(x) &= \max\{|f_i(x)|\}, \quad i = 1,\ldots,21,\\
f_i(x) &= x_1\exp(x_3t_i) + x_2\exp(x_4t_i) - \frac{1}{1+t_i},\\
t_i &= -0.5 + \frac{i-1}{20}.
\end{aligned} \tag{23}$$

For all minimax problems, the search space was $[-50,50]^n$, where n is the dimension of the problem, and the desired accuracy for detecting the global minimizer was 10^{-4}. The swarm size was equal to 20 for Test Problems (TP) 1, 2, 3 and 5, and equal to 50 for the other problems [5]. In the cases where the transformation of Eq. (4) was employed, the value $\alpha_i = 10$ was used for all i [5].

INTEGER PROGRAMMING PROBLEMS

Test Problem 1 [20]. This problem is defined as

$$F_1(x) = \|x\|_1 = |x_1| + \cdots + |x_n|, \tag{24}$$

where n is the dimension. The solution is $x^* = (0,\ldots,0)^\top$ with $F_1(x^*) = 0$. We considered this problem for $n = 30$.

Test Problem 2 [20]. This problem is defined as

$$F_2(x) = x^\top x = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \tag{25}$$

where n is the dimension. The solution is $x^* = (0,\ldots,0)^\top$ with $F_2(x^*) = 0$.

Test Problem 3 [21]. This problem is 5–dimensional and it is defined as

$$F_3(x) = -\begin{pmatrix} 15 & 27 & 36 & 18 & 12 \end{pmatrix}x + x^\top\begin{pmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{pmatrix}x. \tag{26}$$

The best known solutions are $x^* = (0,11,22,16,6)^\top$ and $x^* = (0,12,23,17,6)^\top$, with $F_3(x^*) = -737$.

Test Problem 4 [21]. This problem is 2–dimensional and it is defined as

$$F_4(x) = (9x_1^2 + 2x_2^2 - 11)^2 + (3x_1 + 4x_2^2 - 7)^2. \tag{27}$$

Its solution is $x^* = (1,1)^\top$ with $F_4(x^*) = 0$.

Test Problem 5 [21]. This problem is 4–dimensional and it is defined as

$$F_5(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4. \tag{28}$$

Its solution is $x^* = (0,0,0,0)^\top$ with $F_5(x^*) = 0$.

Test Problem 6 [22]. This problem is 2–dimensional and it is defined as

$$F_6(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2. \tag{29}$$

Its solution is $x^* = (2,-1)^\top$ with $F_6(x^*) = -6$.

Test Problem 7 [21]. This problem is 2–dimensional and it is defined as

$$F_7(x) = -3803.84 - 138.08x_1 - 232.92x_2 + 123.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2. \tag{30}$$

Its solution is $x^* = (0,1)^\top$ with $F_7(x^*) = -3833.12$.

For all integer programming problems, the search space was $[-100,100]^n$, where n is the dimension of the problem, and the desired accuracy for detecting the global minimizer was 10^{-4}. The swarm size was problem dependent and it was equal to 100 for TP 1, equal to 10 for TP 2 and 6, equal to 70 for TP 3, and equal to 20 for the rest of the problems [6].

In order to avoid deterioration of the algorithms' dynamics, we let the particles assume real values and rounded their components to the nearest integer only for the evaluation of the objective function.

In all cases, the default PSO parameters, $\chi = 0.729$, $c_1 = c_2 = 2.05$, were used [14]. Three UPSO variants were investigated, namely, the main UPSO scheme with $u = 0.2$ and $u = 0.5$, as well as the scheme with mutation of

Table 1. Results for the minimax problems.

TP		PSOg	PSOl	UPSO1	UPSO2	UPSOm
	Suc.	**100**	**100**	**100**	**100**	**100**
1	Mean	2538.20	2831.60	2187.20	2132.00	**1993.80**
	St.D.	1134.02	1076.22	**512.78**	917.50	853.65
	Suc.	**100**	**100**	**100**	**100**	**100**
2	Mean	2140.40	2628.60	2225.60	2153.40	**1775.60**
	St.D.	267.89	367.83	292.95	276.52	**241.94**
	Suc.	**100**	**100**	**100**	**100**	**100**
3	Mean	1585.40	2359.40	1618.40	**1476.60**	1670.40
	St.D.	302.15	551.84	314.39	**268.31**	530.56
	Suc.	**100**	**100**	**100**	**100**	**100**
4	Mean	7073.50	17027.50	7307.50	**6484.00**	12801.50
	St.D.	**1672.71**	4028.42	2098.91	1679.98	5072.11
	Suc.	**100**	**100**	**100**	**100**	**100**
5	Mean	2003.20	2482.80	2040.60	1991.60	**1701.60**
	St.D.	230.23	323.75	253.60	241.78	**184.92**
	Suc.	**100**	**100**	**100**	**100**	**100**
6	Mean	13370.00	36597.50	11081.00	**9845.50**	18294.50
	St.D.	853.72	3458.00	660.11	**497.93**	2389.35
	Suc.	**100**	**100**	**100**	**100**	**100**
7	Mean	**2191.00**	4292.50	3152.00	2687.50	3435.50
	St.D.	**568.20**	1872.03	1239.48	631.39	1487.60
	Suc.	**100**	**100**	**100**	**100**	**100**
8	Mean	5825.50	8852.00	5063.00	**4315.00**	6618.50
	St.D.	3486.59	2546.99	1012.11	**859.57**	2597.54
	Suc.	**100**	**100**	**100**	**100**	**100**
9	Mean	**1666.00**	3373.00	2099.50	1679.50	2128.50
	St.D.	**427.58**	1298.03	660.79	474.72	597.40
	Suc.	79	99	98	93	**100**
10	Mean	5400.63	4336.87	3820.92	5057.53	**3332.50**
	St.D.	6765.86	2376.97	1817.98	7539.99	**1775.41**

Eq. (12) with $u = 0.1$, $r_3 \sim \mathcal{N}(M, \Sigma)$, $M = (0, \ldots, 0)^\top$, $\Sigma = \sigma^2 I$, $\sigma = 0.01$, and I being the identity matrix. These variants were selected due to their good performance in unconstrained and dynamic optimization problems [7, 8]. We will denote these variants as UPSO1, UPSO2, and UPSOm, respectively.

UPSO's performance was compared with the performance of the standard local and global PSO variant, which are denoted as PSOl and PSOg, respectively. For the determination of the search direction $\mathcal{L}_i$ of UPSO, as well as for the local PSO variant, a ring neighborhood topology was used with neighborhood radius equal to 1, in order to take full advantage of its exploration properties.

For each test problem and algorithm, 100 experiments were conducted, recording the success rate of each algorithm, i.e., the number of experiments at which it detected the solution with the desired accuracy, as well as the mean and the standard deviation of the required function evaluations. Moreover, a non–parametric Wilcoxon ranked sum test was performed for each pair of algorithms in order to investigate the statistical significance between their performances in significance level 99%.

In Table 1, the results for the minimax problems are reported with the best value per row being boldfaced, while in Table 2 the statistical hypothesis tests are given, with "+" denoting statistically significant difference between the performances of two algorithms. In all test problems, with the exception of TP 10, all algorithms had a success rate of 100%. In TP 1, 2 and 5, UPSOm (UPSO with mutation) outperformed the other algorithms, having also a statistically significant difference. In TP 3, 4, 6 and 8, UPSO2 (UPSO with $u = 0.5$) had the best performance, followed

Table 2. Wilcoxon ranked sum tests for the minimax problems.

		PSOg	PSOl	UPSO1	UPSO2	UPSOm
	PSOg		+	−	+	+
	PSOl	+		+	+	+
1	UPSO1	−	+		−	+
	UPSO2	+	+	−		+
	UPSOm	+	+	+	+	
	PSOg		+	−	−	+
	PSOl	+		+	+	+
2	UPSO1	−	+		−	+
	UPSO2	−	+	−		+
	UPSOm	+	+	+	+	
	PSOg		+	−	+	−
	PSOl	+		+	+	+
3	UPSO1	−	+		+	−
	UPSO2	+	+	+		+
	UPSOm	−	+	−	+	
	PSOg		+	−	+	+
	PSOl	+		+	+	+
4	UPSO1	−	+		+	+
	UPSO2	+	+	+		+
	UPSOm	+	+	+	+	
	PSOg		+	−	−	+
	PSOl	+		+	+	+
5	UPSO1	−	+		−	+
	UPSO2	−	+	−		+
	UPSOm	+	+	+	+	
	PSOg		+	+	+	+
	PSOl	+		+	+	+
6	UPSO1	+	+		+	+
	UPSO2	+	+	+		+
	UPSOm	+	+	+	+	
	PSOg		+	+	+	+
	PSOl	+		+	+	+
7	UPSO1	+	+		+	−
	UPSO2	+	+	+		+
	UPSOm	+	+	−	+	
	PSOg		+	−	+	+
	PSOl	+		+	+	+
8	UPSO1	−	+		+	+
	UPSO2	+	+	+		+
	UPSOm	+	+	+	+	
	PSOg		+	+	−	+
	PSOl	+		+	+	+
9	UPSO1	+	+		+	−
	UPSO2	−	+	+		+
	UPSOm	+	+	−	+	
	PSOg		−	−	−	+
	PSOl	−		−	+	+
10	UPSO1	−	−		−	+
	UPSO2	−	+	−		−
	UPSOm	+	+	+	−	

Table 3. Results for the integer programming problems.

TP		PSOg	PSOl	UPSO1	UPSO2	UPSOm
	Suc.	75	**100**	**100**	**100**	**100**
1	Mean	20440.00	39128.00	13152.00	**11766.00**	15923.00
	St.D.	6863.24	5128.45	**1705.28**	3804.07	2317.43
	Suc.	**100**	99	**100**	**100**	98
2	Mean	696.10	864.95	677.30	683.20	**618.06**
	St.D.	728.96	189.23	**133.91**	864.77	255.07
	Suc.	97	**100**	**100**	96	**100**
3	Mean	5159.07	20201.30	5818.40	**4996.98**	15556.10
	St.D.	**1498.75**	6826.75	1657.47	1818.31	6929.07
	Suc.	**100**	**100**	**100**	**100**	**100**
4	Mean	353.60	401.20	390.40	370.20	**309.80**
	St.D.	119.59	130.85	108.11	117.71	**89.56**
	Suc.	86	**100**	**100**	**100**	**100**
5	Mean	1652.09	3043.20	1658.00	**1503.20**	1933.20
	St.D.	793.17	783.74	419.85	**372.56**	522.24
	Suc.	**100**	**100**	**100**	**100**	**100**
6	Mean	205.10	241.20	221.50	205.10	**171.50**
	St.D.	61.72	91.72	67.10	66.93	**50.26**
	Suc.	**100**	**100**	**100**	**100**	**100**
7	Mean	425.00	477.40	418.80	404.80	**384.00**
	St.D.	141.12	170.74	137.92	130.80	**87.57**

Table 4. Wilcoxon ranked sum tests for the integer programming problems.

		PSOg	PSOl	UPSO1	UPSO2	UPSOm
	PSOg		+	+	+	+
	PSOl	+		+	+	+
1	UPSO1	+	+		+	+
	UPSO2	+	+	+		+
	UPSOm	+	+	+	+	
	PSOg		+	+	−	−
	PSOl	+		+	+	+
2	UPSO1	+	+		+	+
	UPSO2	−	+	+		−
	UPSOm	−	+	+	−	
	PSOg		+	+	−	+
	PSOl	+		+	+	+
3	UPSO1	+	+		+	+
	UPSO2	−	+	+		+
	UPSOm	+	+	+	+	
	PSOg		+	−	−	+
	PSOl	+		−	−	+
4	UPSO1	−	−		−	+
	UPSO2	−	−	−		+
	UPSOm	+	+	+	+	
	PSOg		+	−	+	+
	PSOl	+		+	+	+
5	UPSO1	−	+		+	+
	UPSO2	+	+	+		+
	UPSOm	+	+	+	+	
	PSOg		+	−	−	+
	PSOl	+		−	−	+
6	UPSO1	−	−		−	+
	UPSO2	−	−	−		+
	UPSOm	+	+	+	+	
	PSOg		−	−	−	−
	PSOl	−		−	+	+
7	UPSO1	−	−		−	−
	UPSO2	−	+	−		−
	UPSOm	−	+	−	−	

by UPSO1 (UPSO with $u = 0.2$), having statistically significant difference between its performance and the performance of the other algorithms. The global variant of the standard PSO (PSOg) proved to be more efficient in TP 7 and 9, although, in the latter, there was no statistically significant difference between its performance and the performance of UPSO2. TP 10 proved to be the most difficult for all algorithms, perhaps due to the large number of functions. The success rate was reduced for all algorithms except UPSOm, which had the best performance. Also, TP 10 was the only case where the standard local PSO (PSOl) outperformed the global variant (PSOg) underlining the difference between their exploration ability, which is crucial in such difficult problems. Summarizing the results for the minimax problems, UPSOm and UPSO2 proved to be the most promising schemes overall, in accordance with results reported for different optimization problems [7, 8]. PSOg proved to be competitive in a few problems, without always having statistically significant difference from UPSO. Finally, in difficult test problems, UPSOm proved to be the most efficient algorithm.

In Tables 3 and 4 the corresponding results and statistical hypothesis tests for the integer programming problems are reported. The rounding of the particles' components to the nearest integer for the evaluation of the objective function seems to introduce a peculiarity in the algorithms' dynamics. Two different real components that are close can be rounded to the same integer, assigning to different particles the same objective function. Obviously, exploitation–based schemes can be affected by this procedure, since they are more prone to move the particles faster towards the best positions detected by the swarm. Thus, schemes that promote exploration were expected to have better performance. This is indeed reflected in the results reported in Table 3. PSOl (local PSO variant) had better success rates than PSOg (global variant) in all but the last test problem. Also, UPSO1 ($u = 0.2$), which promotes exploration rather than exploitation, was more competitive to the "balanced" UPSO2 scheme ($u = 0.5$) than in the minimax problems, having the best performance in TP 2 and 3, while UPSO2 performed better in TP 1 and 5. UPSOm had the best performance in the other problems, although it did not have statistically significant difference in the last test problem.

In general, both in minimax and integer programming problems, the standard UPSO variants seemed to follow the performance pattern of PSO, i.e., in problems where PSOl has a good performance, UPSO schemes that promote exploration are more efficient, while in the other cases more balanced schemes, such as UPSO2, proved to have better performance. The overall good performance of UPSO with

mutation indicates that properly perturbed exploration–based schemes of UPSO retain a good balance between exploration and exploitation. This behavior is in accordance with previously reported results in different static and dynamic optimization problems [7, 8], thereby rendering UPSOm a good default choice.

4. CONCLUSIONS

We investigated the performance of Unified Particle Swarm Optimization on minimax and integer programming problems. Different variants of the algorithm were investigated and compared with the standard Particle Swarm Optimization algorithm. The results indicate that UPSO can tackle efficiently problems of both types, outperforming in most cases the standard PSO. Useful conclusions were also derived regarding the performance of different UPSO variants, as well as on the effect of rounding on the algorithms' performance in integer programming problems.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their useful comments and suggestions. This work was partially supported by the PENED 2001 Project awarded by the Greek Secretariat of Research and Technology.

5. REFERENCES

[1] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. McGraw–Hill, 1995.

[2] G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors. *Handbooks in OR & MS*, volume 1. Elsevier, 1989.

[3] D. Z. Du and P. M. Pardalos. *Minimax and Applications*. Kluwer, 1995.

[4] J. W. Bandler and C. Charalambous. Nonlinear programming using minimax techniques. *J. Optim. Th. Appl.*, 13:607–619, 1974.

[5] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis. Particle swarm optimization for minimax problems. In *Proceedings of the IEEE 2002 Congress on Evolutionary Computation*, pages 1582–1587, Hawaii (HI), USA, 2002. IEEE Press.

[6] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis. Particle swarm optimization for integer programming. In *Proceedings of the IEEE 2002 Congress on Evolutionary Computation*, pages 1582–1587, Hawaii (HI), USA, 2002. IEEE Press.

[7] K. E. Parsopoulos and M. N. Vrahatis. UPSO: A unified particle swarm optimization scheme. In *Lecture Series on Computer and Computational Sciences, Vol. 1, Proceedings of the International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004)*, pages 868–873. VSP International Science Publishers, Zeist, The Netherlands, 2004.

[8] K. E. Parsopoulos and M. N. Vrahatis. Unified particle swarm optimization in dynamic environments. *Lecture Notes in Computer Science (LNCS)*, 3449:590–599, 2005.

[9] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings Sixth Symposium on Micro Machine and Human Science*, pages 39–43, Piscataway, NJ, 1995. IEEE Service Center.

[10] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE Int. Conf. Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[11] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[12] J. Kennedy. Bare bones particle swarms. In *Proc. IEEE Swarm Intelligence Symposium*, pages 80–87, Indianapolis, USA, 2003. IEEE Press.

[13] R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: Simpler, maybe better. *IEEE Trans. Evol. Comput.*, 8(3):204–210, 2004.

[14] M. Clerc and J. Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.*, 6(1):58–73, 2002.

[15] I. C. Trelea. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, 85:317–325, 2003.

[16] J. Matyas. Random optimization. *Automatization and Remote Control*, 26:244–251, 1965.

[17] S. Xu. Smoothing method for minimax problems. *Comp. Optim. Appl.*, 20:267–279, 2001.

[18] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[19] L. Lukšan and J. Vlček. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report 798, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 2000.

[20] G. Rüdolph. An evolutionary algorithm for integer programming. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, volume 3, pages 139–148. Springer, 1994.

[21] A. Glankwahmdee, J. S. Liebman, and G. L. Hogg. Unconstrained discrete nonlinear programming. *Engineering Optimization*, 4:95–107, 1979.

[22] S. S. Rao. *Engineering Optimization–Theory and Practice*. Wiley, 1996.