Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational and Applied Mathematics 175 (2005) 113–136
JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS
www.elsevier.com/locate/cam

Computing Nash equilibria through computational intelligence methods

N.G. Pavlidis, K.E. Parsopoulos, M.N. Vrahatis*

Department of Mathematics, University of Patras Artificial Intelligence Center (UPAIRC), University of Patras, GR 26110 Patras, Greece

Received 5 October 2003; received in revised form 18 February 2004

Abstract

Nash equilibrium constitutes a central solution concept in game theory. The task of detecting the Nash equilibria of a finite strategic game remains a challenging problem up-to-date. This paper investigates the effectiveness of three computational intelligence techniques, namely, covariance matrix adaptation evolution strategies, particle swarm optimization, as well as, differential evolution, to compute Nash equilibria of finite strategic games, as global minima of a real-valued, nonnegative function. An issue of particular interest is to detect more than one Nash equilibria of a game. The performance of the considered computational intelligence methods on this problem is investigated using multistart and deflection.

MSC: 91A99; 91A05; 91A06; 74P99

Keywords: Nash equilibria; Evolutionary algorithms; Particle swarm optimization; Differential evolution; Evolution strategies

1. Introduction

Game theory is a mathematical theory of socio-economic phenomena exhibiting interaction among decision-makers, called *players*, whose actions affect each other. The fundamental assumptions that underlie the theory are that players pursue well-defined exogenous objectives and take into account

* Corresponding author. Tel.: +30-2610-997374; fax: +30-2610-992965.
E-mail addresses: npav@math.upatras.gr (N.G. Pavlidis), kostasp@math.upatras.gr (K.E. Parsopoulos), vrahatis@math.upatras.gr (M.N. Vrahatis).

0377-0427/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2004.06.005

their knowledge, or expectations, of other players' behavior [17]. The theory has been so far applied in the fields of economics, political science, evolutionary biology, computer science, statistics, accounting, social psychology, law, and branches of philosophy such as epistemology and ethics [1].

A *game* is a model of strategic interaction among a number of players, which includes the constraints on the actions that players can take and the players' interests, but does not specify the actions that players do take. A *solution* is a systematic description of the outcomes that may emerge in a game [17]. In this paper we consider only the family of *strategic*, or *normal form*, games. The most commonly encountered solution concept in game theory is that of *Nash equilibrium* [15,16]. This notion captures a steady state of the play of a strategic game, in which each player holds correct expectations concerning the other players' behavior and acts rationally.

The problem of detecting the Nash equilibria of a finite strategic game admits a number of alternative formulations, yet computing such solutions remains a challenging task up-to-date (for a comprehensive review on algorithms to compute equilibria of *n*-person games see [13], for a survey of algorithms for 2-player games see [28]). Furthermore, an algorithm that computes a single Nash equilibrium is unsatisfactory for many applications. Even if the resulting equilibrium is perfect, or satisfies some other criterion posed in the literature on refinements of Nash equilibrium, we cannot eliminate the possibility that other, more salient equilibria exist [13].

The problem of computing a Nash equilibrium can be formulated as a global optimization problem [12]. This formulation allows us to consider three computational intelligence methods, namely, covariance matrix adaptation evolution strategies (CMA-ES), particle swarm optimization (PSO), and differential evolution (DE), to detect Nash equilibria. CMA-ES, PSO and DE are stochastic optimization methods capable of handling nondifferentiable, nonlinear and multimodal objective functions. They exploit a population of potential solutions to probe the search space synchronously. Each member of the population adapts its position towards the most promising regions of the function's landscape, characterized, in the case of minimization, by lower function values. Incorporating *multistart* [30] or *deflection* [11], more than one global minima of the objective function can be obtained.

The remaining paper is organized as follows: Section 2 is devoted to a brief exposition of basic concepts of game theory and the formulation of the problem. In Section 3, the computational intelligence methods considered, as well as, the techniques employed to compute more than one minimizers of a function, are described. Section 4 outlines the proposed algorithmic scheme and discusses the experimental results. The paper ends with conclusions in Section 5.

2. Notation and problem formulation

2.1. Strategic games and Nash equilibrium

A *finite strategic game*, $\Gamma = \langle(\mathcal{N}), (S_i), (u_i)\rangle$, is defined by [17],

- a finite set $\mathcal{N} = \{1, \ldots, N\}$ of *players*,
- for each player $i \in \mathcal{N}$, a set of actions, pure strategies, $S_i = \{s_{i1}, \ldots, s_{im_i}\}$,
- for each player $i \in \mathcal{N}$, a *payoff function*, $u_i : S \to \mathbb{R}$, is also defined, where $S = S_1 \times S_2 \times \ldots \times S_{\mathcal{N}}$ is the Cartesian product of all sets S_i.

Next, we give the notation that will be used in the paper. The following is based on [13]. Let $\mathscr{P}_i$ be the set of real valued functions on S_i. The notation $p_{ij} = p_i(s_{ij})$, is used for the elements $p_i \in \mathscr{P}_i$. Let also $\mathscr{P} = \times_{i\in\mathscr{N}} \mathscr{P}_i$ and $m = \sum_{i\in\mathscr{N}} m_i$. Then $\mathscr{P}$ is isomorphic to $\mathbb{R}^m$. We denote elements in $\mathscr{P}$ by $p = (p_1, p_2, \ldots, p_N)$, where $p_i = (p_{i1}, p_{i2}, \ldots, p_{im_i}) \in \mathscr{P}_i$. If $p \in \mathscr{P}$, and $p_i' \in \mathscr{P}_i$, then (p_i', p_{-i}) stands for the element $q \in \mathscr{P}$ that satisfies, $q_i = p_i'$ and $q_i = p_j$ for $j \neq i$.

Now let Δ_i be the set of probability measures on S_i. We define $\Delta = \times_{i\in\mathscr{N}} \Delta_i$, so $\Delta \subseteq \mathbb{R}^m$. Thus, the elements $p_i \in \Delta_i$ are real valued functions on S_i, $p_i : S_i \to \mathbb{R}$ and it holds that,

$$\sum_{s_{ij}\in S_i} p_i(s_{ij}) = 1, \quad p_i(s_{ij}) \geqslant 0, \quad \forall s_{ij} \in S_i.$$

We use the abusive notation s_{ij} to denote the strategy $p_i \in \Delta_i$ with $p_{ij} = 1$. Hence, the notation (s_{ij}, p_{-i}) represents the strategy where player i adopts the pure strategy s_{ij}, and all the other players adopt their components of p.

The payoff function u is extended to have domain $\mathbb{R}^m$ by the rule,

$$u_i(p) = \sum_{s\in S} p(s)u_i(s), \tag{1}$$

$$p(s) = \prod_{i\in\mathscr{N}} p_i(s_i). \tag{2}$$

Definition 1. A strategy profile, $p^* = (p_1^*, p_2^*, \ldots, p_N^*) \in \Delta$ is a *Nash equilibrium* if $p^* \in \Delta$ and for all $i \in \mathscr{N}$ and all $p_i \in \Delta_i$, $u_i(p_i, p_{-i}^*) \leqslant u_i(p^*)$.

An immediate implication of the above definition is that for a strategy profile p^* to be a Nash equilibrium, it must be that no player i has an action yielding a payoff that he strictly prefers to the payoff he receives by choosing p_i^*, assuming that every other player j chooses his equilibrium action p_j^*. In other words, no player can profitably deviate, given the actions of other players.

2.2. Problem formulation

As previously mentioned, the problem of finding a Nash equilibrium of a normal form game can be formulated as a problem of detecting the global minimum of a real valued function [12]. To this end, three functions, x, z and $g : \mathscr{P} \to \mathbb{R}^m$, are defined. For any $p \in \mathscr{P}$, $i \in \mathscr{N}$ and $s_{ij} \in S_i$, define the ijth component as,

$$x_{ij}(p) = u_i(s_{ij}, p_{-i}), \tag{3}$$

$$z_{ij}(p) = x_{ij}(p) - u_i(p), \tag{4}$$

$$g_{ij}(p) = \max[z_{ij}(p), 0]. \tag{5}$$

Now, we define the real valued function $v : \Delta \to \mathbb{R}$, by,

$$v(p) = \sum_{i\in\mathscr{N}} \sum_{1\leqslant j\leqslant m_i} [g_{ij}(p)]^2. \tag{6}$$

Function v is continuous, differentiable, and satisfies the inequality $v(p) \geqslant 0$, for all $p \in \Delta$. Furthermore, p^* is a Nash equilibrium, if and only if, it is a global minimum of v, i.e. $v(p^*) = 0$ [12,13].

3. Computational intelligence methods considered

3.1. Covariance matrix adaptation evolution strategies

Evolution strategies (ES) are population-based search algorithms developed by Rechenberg and Schwefel [22–25]. ES exploit a population of μ individuals to probe the search space. At each iteration of the algorithm, λ offsprings are produced by stochastic variation, called mutation, of recombinations of a set of individuals (called the *parents*) from the current population. Mutation is typically carried out by adding a realization of a normally distributed random vector. After the creation of the offspring individuals, a selection phase takes place, where either the μ best individuals among the offspring population, or the μ best individuals among both the parent and the offspring populations are selected to form the population of the next generation. These two selection schemes are denoted as (μ, λ)-ES and $(\mu+\lambda)$-ES, respectively.

ES use a set of parameters, called *strategy parameters*, to parameterize the normal distribution used in the mutation procedure. These parameters can either be fixed, or evolve during the evolution process resulting in self-adaptive ES [2]. Clearly, the parameters of the normal distribution play an important role in the performance of the ES algorithm [7]. The adaptation of the strategy parameters in ES usually takes place within the concept of *mutative strategy parameter control* (MSC). In this context, strategy parameters are mutated and search points are subsequently generated by means of this mutated strategy parameter setting.

Covariance matrix adaptation evolution strategies (CMA-ES) have been developed by Hansen and Ostermeier [6,7]. CMA-ES explicitly realize the objective of MSC, which is to favor strategy parameter settings that produce individuals that are selected (in the minimization framework, this implies individuals with lower function values). Instead of utilizing selection information from a single generation step, CMA-ES utilize a whole path taken by the population over a number of generations. Hansen and Ostermeier call such paths *evolution paths*. If successively selected mutation steps are parallel correlated, the evolution path will be comparatively long, and vice versa if successively selected mutation steps are anti–parallel correlated. An evolution path is calculated through an iterative process by weighted summation of successively selected mutation steps (cf. Eq. (9)). Moreover, CMA-ES implements a principal component analysis of the previously selected mutation steps to determine the new mutation distribution. An advantage of this approach is that it renders the adaptation mechanism inherently independent of the coordinate system [7].

The proposed scheme for CMA-ES is $(\mu_{\mathrm{W}}, \lambda)$, where μ_{W} denotes weighted recombination from all μ individuals of the parent population. CMA-ES exploit a set of parameters, $p_{c,G} \in \mathbb{R}^n$, $C_G \in \mathbb{R}^{n\times n}$, $p_{\sigma,G} \in \mathbb{R}^n$, and $\sigma_G \in \mathbb{R}^+$, where G denotes the generation number. The parameters are initialized as follows: $p_{c,0}=p_{\sigma,0}=0$ and $C_0=I$ (the unity matrix), while σ_0 and the initial weighted mean of the μ parent individuals, $\langle X\rangle_{\mathrm{W},0}$, have to be chosen problem dependent [7]. The offsprings, $X_{k,G+1}$, $k=1,\ldots,\lambda$, are then determined by the equation,

$$X_{k,G+1} = \langle X\rangle_{\mathrm{W},G} + \sigma_G \underbrace{B_G D_G z_{k,G+1}}_{\sim\mathcal{N}(0,C_G)},$$

where $\mathcal{N}(0, C_G)$ denotes the Gaussian distribution with zero mean and covariance matrix C_G, and,

$$\langle X\rangle_{\mathrm{W},G} = \frac{1}{\sum_{i=1}^{\mu} w_i} \sum_{i=1}^{\mu} w_i X_{i:\lambda,G}, \quad w_i \in \mathbb{R}^+,$$

with $i:\lambda$ denoting the ith best individual, is the weighted mean of the μ best individuals at generation G; $\sigma_G \in \mathbb{R}^+$ is the step size; $z_{k,G+1} \in \mathbb{R}^n$ are independent realizations of a $(0, I)$-normally distributed random vector. The matrices B_G and D_G are defined by the symmetrical positive definite $n \times n$ covariance matrix C_G, as follows [7],

$$C_G = B_G D_G (B_G D_G)^{\top} = B_G (D_G)^2 B_G^{\top}.$$

This is actually a singular value decomposition of C_G. Thus, the matrix D_G is an $n \times n$ diagonal matrix with its diagonal elements being equal to the square roots of the eigenvalues of C_G, while, B_G is an $n \times n$ orthogonal matrix that determines the coordinate system, where the scaling with D_G takes place and its columns are the normalized eigenvectors of C_G. The matrix C_G is updated by means of the evolution path $p_{c,G+1}$,

$$p_{c,G+1} = (1 - c_c) p_{c,G} + c_c^u c_w B_G D_G \langle z\rangle_{\mathrm{W},G+1}, \tag{7}$$

$$C_{G+1} = (1 - c_{\mathrm{cov}}) C_G + c_{\mathrm{cov}} p_{c,G+1} p_{c,G+1}^{\top}, \tag{8}$$

where, $p_{c,G}$ stands for the weighted differences of points $\langle x\rangle_{\mathrm{W}}$; $c_c \in [0, 1]$ determines the cumulation time for p_c; $c_c^u = \sqrt{c_c(2 - c_c)}$ normalizes the variance of the p_c,

$$c_{\mathrm{W}} = \frac{\sum_{i=1}^{\mu} w_i}{\sqrt{\sum_{i=1}^{\mu} w_i^2}}, \quad \langle z\rangle_{\mathrm{W},G+1} = \frac{1}{\sum_{i=1}^{\mu} w_i} \sum_{i=1}^{\mu} w_i z_{i:\lambda,G+1},$$

and $c_{\mathrm{cov}} \in [0, 1]$ determines the change rate of the matrix C. The evolution of the global step-size σ_G is determined by a "conjugate" evolution path, $p_{\sigma,G+1}$,

$$p_{\sigma,G+1} = (1 - c_\sigma) p_{\sigma,G} + c_\sigma^u c_{\mathrm{W}} B_G \langle z\rangle_{\mathrm{W},G+1}, \tag{9}$$

$$\sigma_{G+1} = \sigma_G \exp\left(\frac{1}{d_\sigma} \frac{\| p_{\sigma,G+1} \| - \hat{\chi}_n}{\hat{\chi}_n}\right), \tag{10}$$

where, c_σ determines the cumulation time; $c_\sigma^u = \sqrt{c_\sigma(2 - c_\sigma)}$; d_σ is a damping parameter that affects the feasible change rate of σ_G; and $\hat{\chi}_n$ represents the expectation of the length of a $(0, I)$-normally distributed random vector.

3.2. Particle swarm optimization

Particle swarm optimization (PSO) belongs to the class of swarm intelligence algorithms. The ideas that underlie PSO are inspired not by the evolutionary mechanisms encountered in natural selection, but rather by the social behavior of flocking organisms, such as swarms of birds and fish schools. It has been observed that the behavior of the individuals that comprise a flock adheres to fundamental rules like nearest-neighbor velocity matching and acceleration by distance [9,10]. In this respect, it has been claimed that PSO performs mutation with a conscience [10].

PSO is a population-based algorithm that exploits a population of individuals, to synchronously probe promising regions of the search space. In this context, the population is called a *swarm*, and the individuals are called *particles*. Each particle moves with an adaptable velocity within the search space, and retains in its memory the best position it ever encountered. In the *global* variant of PSO the best position ever attained by all individuals of the swarm is communicated to all the particles. In the *local* variant, each particle is assigned to a neighborhood consisting of a prespecified number of particles. In this case, the best position ever attained by the particles that comprise the neighborhood is communicated among them [10,19].

Assume an n-dimensional search space, $S \subset \mathbb{R}^n$, and a swarm consisting of NP particles. The ith particle is in effect an n-dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top$. The velocity of this particle is also an n-dimensional vector, $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top$. The best previous position encountered by the ith particle is a point in S, denoted as $\mathrm{BP}_i = (bp_{i1}, bp_{i2}, \ldots, bp_{in})^\top$. Assume g to be the index of the particle that attained the best previous position among all the individuals of the swarm, and G to be the iteration counter.

Then, according to the *constriction factor* version of PSO, the velocity of the ith particle of the swarm is determined by the following equation [3],

$$V_{i,G+1} = \chi[V_{i,G} + c_1 r_1(\mathrm{BP}_{i,G} - X_{i,G}) + c_2 r_2(\mathrm{BP}_{g,G} - X_{i,G})], \tag{11}$$

where $i = 1, 2, \ldots, \mathrm{NP}$; χ is the constriction factor; c_1 and c_2 are called the *cognitive* and *social* parameters, respectively; and r_1, r_2 are random numbers uniformly distributed in the interval $[0, 1]$. Alternatively, the update of the velocity of the particle can be performed through the *inertia weight* variant of the algorithm [4,26,27],

$$V_{i,G+1} = wV_{i,G} + c_1 r_1(\mathrm{BP}_{i,G} - X_{i,G}) + \ c_2 r_2(\mathrm{BP}_{g,G} - X_{i,G}), \tag{12}$$

where w is called the *inertia weight*. The position of the ith particle in iteration $G + 1$ is computed by,

$$X_{i,G+1} = X_{i,G} + V_{i,G+1}. \tag{13}$$

Both the constriction factor, χ, and the inertia weight, w, are mechanisms for the control of the magnitude of velocities. However, there are some major differences regarding the way these two are computed and applied. The constriction factor is derived analytically through the formula [3],

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \tag{14}$$

for $\phi > 4$, where $\phi = c_1 + c_2$, and $\kappa = 1$. Different configurations of χ, as well as a thorough theoretical analysis of the derivation of Eq. (14), can be found in [3,31]. The inertia weight, w, in Eq. (12), is employed to manipulate the impact of the previous history of velocities on the current velocity. Therefore, w resolves the trade-off between the global and local exploration ability of the swarm. A large inertia weight encourages global exploration (visiting unexplored areas of the search space), while a small one promotes local exploration, i.e. probing the current search area. A suitable value for w provides the desired balance between the global and local exploration ability of the swarm, and consequently improves the effectiveness of the algorithm. Therefore, it is preferable to initialize the inertia weight to a large value,

giving priority to global exploration of the search space, and gradually decrease it, so as to obtain refined solutions [26,27]. This finding is intuitively very appealing. In conclusion, an initial value of w around 1.0 and a gradual decline towards 0 is considered a proper choice for w. The initialization of the swarm and the velocities, is usually performed randomly in the search space, although more sophisticated initialization techniques can enhance the overall performance of the algorithm [18].

Thorough theoretical investigations of the convergence properties of PSO through analyzing the trajectories of the particles are provided in [3,31]. These studies are based on analyzing initially a simplified deterministic model of the algorithm in order to provide an understanding about how it probes the search space, and then continue on to analyze the full stochastic system [3,31]. Generalized models of the algorithm are proposed, and techniques for controlling the convergence properties of the particle system through the fine-tuning of the parameters are analyzed in [3,31].

3.3. Differential evolution

Storn and Price [29] introduced a novel minimization method, called differential evolution (DE), capable of handling nondifferentiable, nonlinear and multimodal objective functions. DE exploits a *population* of NP potential solutions, called *individuals*, that are n-dimensional vectors, to probe the search space. At each generation, G, of the algorithm three operators, namely, *mutation*, *recombination* and *selection*, are performed in order to obtain more accurate approximations to a solution [21]. All individuals are uniformly initialized in the search space. At the mutation step, for each $i = 1, \ldots, \mathrm{NP}$, a new mutant vector $V_{i,G+1}$ is generated by combining a number of vectors from the population of the current generation. Specifically, for each individual $X_{i,G}$, $i = 1, \ldots, \mathrm{NP}$, a new individual $V_{i,G+1}$ (mutant individual) is generated according to one of the following equations,

$$V_{i,G+1}=X_{\mathrm{best},G} + Q(X_{r_1,G} - X_{r_2,G}), \tag{15}$$
$$V_{i,G+1}=X_{r_1,G} + Q(X_{r_2,G} - X_{r_3,G}), \tag{16}$$
$$V_{i,G+1}=X_{i,G} + Q(X_{\mathrm{best},G} - X_{i,G}) + Q(X_{r_1,G} - X_{r_2,G}), \tag{17}$$
$$V_{i,G+1}=X_{\mathrm{best},G} + Q(X_{r_1,G} - X_{r_2,G}) + Q(X_{r_3,G} - X_{r_4,G}), \tag{18}$$
$$V_{i,G+1}=X_{r_1,G} + Q(X_{r_2,G} - X_{r_3,G}) + Q(X_{r_4,G} - X_{r_5,G}), \tag{19}$$
$$\begin{aligned} V_{i,G+1}=&(X_{r_1,G} + X_{r_2,G} + X_{r_3,G})/3 + (p_2 - p_1)(X_{r_1,G} - X_{r_2,G}) \\ &+ (p_3 - p_2)(X_{r_2,G} - X_{r_3,G}) + (p_1 - p_3)(X_{r_3,G} - X_{r_1,G}), \end{aligned} \tag{20}$$

where $X_{\mathrm{best},G}$ denotes the best individual of the previous generation; $Q > 0$ is a real parameter, called *mutation constant*, which controls the amplification of the difference between two individuals so as to avoid the stagnation of the search process; and $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, \ldots, \mathrm{NP}\}$, are random integers mutually different and different from the running index i. The mutation strategy of Eq. (20) is known as the *trigonometric mutation strategy*, and has been recently proposed in [5]. This strategy performs a mutation according to Eq. (20) with probability τ_Q and a mutation according to Eq. (16) with probability $(1 - \tau_Q)$. The values of $p_i, i \in \{1, 2, 3\}$ and p' are obtained through the following equations,

$$\begin{aligned} p_1&=|f(X_{r_1,G})|/p', \\ p_2&=|f(X_{r_2,G})|/p', \\ p_3&=|f(X_{r_3,G})|/p', \\ p'&=|f(X_{r_1,G})| + |f(X_{r_2,G})| + |f(X_{r_3,G})|. \end{aligned}$$

At this point it is worth noting that performance differences caused by the selection of the mutation strategy for DE will be provided in Section 4.

The resulting mutant vectors are mixed with a predetermined vector, called *target* vector. This operation is called *recombination* (crossover), and it gives rise to the *trial* vector. At the recombination step, for each component $j = 1, 2, ..., n$, of the mutant vector, a random number $r \in [0, 1]$ is generated. If r is smaller than the predefined *recombination constant*, $\mathrm{CR} \in [0, 1]$, the jth component of the mutant vector $V_{i,G+1}$ becomes the jth component of the trial vector. Otherwise, the jth component of the target vector, $X_{i,G}$, is selected as the jth component of the trial vector, which is defined by,

$$U_{i,G+1} = (u_{i1,G+1}, u_{i2,G+1}, \ldots, u_{in,G+1}),$$

where

$$u_{ij,G+1} = \begin{cases} v_{ij,G+1}, & \text{if } (\mathrm{randb}(j) \leqslant \mathrm{CR}) \text{ or } j = \mathrm{rnbr}(i), \\ x_{ij,G}, & \text{if } (\mathrm{randb}(j) > \mathrm{CR}) \text{ and } j \neq \mathrm{rnbr}(i), \end{cases}$$

where $j = 1, 2, \ldots, n$; randb(j) is the jth evaluation of a uniform random number generator within the range [0, 1]; and rnbr(i) is a randomly chosen index from the set $\{1, 2, \ldots, n\}$.

Finally, at the selection step, the trial vector obtained after the recombination step is accepted for the next generation, if and only if, it yields a reduction of the value of the objective function, $f(\cdot)$, relative to the previous vector, $X_{i,G}$; if not, $X_{i,G}$ is retained,

$$X_{i,G+1} = \begin{cases} U_{i,G+1}, & \text{if } f(U_{i,G+1}) < f(X_{i,G}), \\ X_{i,G}, & \text{otherwise.} \end{cases}$$

3.4. Detecting more than one minimizers

The most simple technique to compute more than one minimizers of a function is *multistart* [30]. In this approach, as soon as a minimizer is detected the algorithm is reinitialized in the search space. This approach, however, does not guarantee that the algorithm will not converge to one of the previously detected minimizers.

The *deflection* technique, proposed in [11], is an alternative technique that allows multiple minimizers to be obtained in a single run of an optimization algorithm. Let $f : S \to \mathbb{R}$, $S \subset \mathbb{R}^n$, be the original objective function under consideration. Let also x_i^*, $i = 1, \ldots, m$, be m minimizers of f. Then, the *deflection* technique defines a new function $F(x)$ as follows,

$$F(x) = T_1(x; x_1, \lambda_1)^{-1} \cdots T_m(x; x_m, \lambda_m)^{-1} f(x), \tag{21}$$

where λ_i, $i = 1, \ldots, m$, are relaxation parameters, and $T_1, \ldots, T_m$, are appropriate functions in the sense that the resulting function F has exactly the same minimizers as f, except at points $x_1^*, \ldots, x_m^*$. The functions,

$$T_i(x; x_i, \lambda_i) = \tanh(\lambda_i \parallel x - x_i^* \parallel), \quad i = 1, \ldots, m, \tag{22}$$

satisfy this property, known as the *deflection property*, as shown in [11]. Therefore, when the optimization algorithm detects a minimizer, x_i^*, of the objective function, the algorithm is restarted and an additional $T_i(x; x_i, \lambda_i)$ is included in the objective function $F(x)$.

Alternative configurations of the parameter λ result in different shapes of the transformed function. For larger values of λ the effect of the deflection technique on the objective function is relatively mild. On the other hand, using $\lambda < 1$ results in a function F with considerably larger function values in the neighborhood of the deflected minimizer. Deflection has been effectively used with PSO for detecting periodic orbits of nonlinear mappings [20].

4. The proposed approach and experimental results

4.1. Proposed algorithm

The proposed algorithm for detecting several Nash equilibria, can be summarized in the following three steps:

Step 1: Apply one of the aforementioned optimization methods to detect a global minimizer (Nash equilibrium) of the objective function.
Step 2: Once a global minimizer is detected store it.
Step 3: If the number of restarts allowed is not exceeded, apply multistart or deflection and go to Step 1. Otherwise, terminate the algorithm and report the results.

4.2. Test problems

The performance of the algorithm on finding Nash equilibria, was studied on six benchmark problems which are included in the latest stable version (ver. 0.97.0.5) of the state-of-the-art GAMBIT software suite [14] (GAMBIT is freely available from `http://econweb.tamu.edu/gambit/`). All games were characterized by more than one Nash equilibria. In all problems, the main goal was to detect several (and if possible all the) equilibria. To obtain the list of Nash equilibria of each game, the GAMBIT routine `PolEnumSolve` was used. Next, the test problems used are reported. For games with more than three players the payoff matrices are not reported due to space limitations. The name of the GAMBIT file that corresponds to each game is mentioned so that the reader can obtain all the information about these games from GAMBIT.

Test Problem 1. This is a four-person, normal form game, with 2 pure strategies available to each player. The game is characterized by three equilibria. The GAMBIT file that corresponds to this game is named `2x2x2x2.nfg`.

Test Problem 2. This is another four-person, normal form game, with 2 pure strategies available to each player. The game is characterized by five mixed equilibria. The GAMBIT file that corresponds to this game is named `g3.nfg`.

Test Problem 3. This is a five player game, with two pure strategies available to each player. The game is characterized by five Nash equilibria. The GAMBIT file that corresponds to this game is named `2x2x2x2x2.nfg`.

Test Problem 4. This is a normal form game with three players and two pure strategies available to each player [12]. The payoffs of this game are given in Table 1. This game has a total of 9 Nash equilibria,

Table 1
Payoff matrices for Test Problem 4

s_{31}	s_{21}	s_{22}	s_{32}	s_{21}	s_{22}
s_{11}	9, 8, 12	0, 0, 0	s_{11}	0, 0, 0	3, 4, 6
s_{12}	0, 0, 0	9, 8, 2	s_{12}	3, 4, 4	0, 0, 0

Table 2
Payoff matrix for Test Problem 6

	s_{21}	s_{22}	s_{23}	s_{24}
s_{11}	3, 2	0, 0	0, 0	0, 0
s_{12}	0, 0	2, 2	0, 0	0, 0
s_{13}	0, 0	0, 0	1, 4	0, 0
s_{14}	0, 0	0, 0	0, 0	4, 7

Table 3
Number of restarts for each test problem

Test problem	1	2	3	4	5	6
Restarts	8	10	10	15	18	20

four pure strategy equilibria, three mixed strategy equilibria, and and two full support strategy equilibria. The GAMBIT file that corresponds to this game is `2x2x2.nfg`.

Test Problem 5. This is a three player coordination game with three strategies available to each player. The game is characterized by 13 equilibria. The GAMBIT file that corresponds to this game is `coord333.nfg`.

Test Problem 6. This is a two-player game with four strategies available to each player. This game has a total of 15 Nash equilibria. The GAMBIT file that corresponds to this game is `coord4.nfg` (Table 2).

4.3. Experimental setup

Numerical experiments were performed using a DE, a PSO, and a CMA-ES, C++ Interface developed under the Fedora Linux 1.0 operating system using the GNU compiler collection (gcc) version 3.3.2. For the singular value decomposition performed by the CMA-ES algorithm the C Linear Algebra PACKage (CLAPACK) 3.0 and the Automatically Tuned Linear Algebra Software (ATLAS) 3.4.2 libraries were used.

For each test problem, all the algorithms were allowed to perform a number of restarts depending on the number of Nash equilibria of the game under consideration. Each algorithm was allowed to perform a prespecified number of iterations per restart. The stopping criterion employed was to achieve a function value less than or equal to 10^{-8}. Otherwise, the algorithm was reinitialized when the maximum number of iterations per restart was reached. The number of restarts for each test problem is illustrated in Table 3. When the deflection technique was used, the values of the relaxation parameters, λ_i of Eq. (21), were set to 1.

Table 4
Default parameter setting for the CMA-ES algorithm [7]

λ	μ	$w_{i=1,\ldots,\mu}$	c_c	c_{cov}	c_σ	d_σ
$4 + \lfloor 3\ln(n)\rfloor$	$\lfloor\lambda/2\rfloor$	$\ln(\frac{\lambda+1}{2}) - \ln(i)$	$\frac{4}{n+4}$	$\frac{2}{(n+\sqrt{2})^2}$	$\frac{4}{n+4}$	$c_\sigma^{-1} + 1$

Table 5
Population size and iterations per restart for DE and PSO

Problem	Pop. size	Iterations
TP1	20	1000
TP2	20	1000
TP3	50	2000
TP4	10	1000
TP5	20	1000
TP6	10	1000

For each point, X, to be a Nash equilibrium of a game it must be a minimizer of Eq. (6) and $X \in \Delta$, as defined in Section 2. To evaluate the function value of each individual, X, we use the following normalization:

$$x_{ij}^p = \frac{\| x_{ij} \|}{\sum_{j=1}^{m_i} \| x_{ij} \|},$$

with $i \in \mathcal{N}$, and $j = 1, \ldots, m_i$, which ensures that $X^p \in \Delta$. Note that this normalization is used only to compute the objective function of Eq. (6) and not to constrain the populations to lie in Δ. Our experience indicates that if the normalized individuals, X^p, replace the original individuals, X, then the diversity of the population decreases drastically and this in turn causes the premature convergence of the considered methods.

For the CMA-ES, the default parameter setup suggested in [7] was adopted. This setup is illustrated in Table 4. The initial component-wise standard deviation of the mutation step, σ_0, was set to 1.0 for all games. Moreover, the maximum number of generations per restart was set to 1000.

For the DE algorithm, the values for the mutation constant, Q, and the recombination constant CR were set to 0.7 and 0.9, respectively, following the suggestions in [29]. Concerning the trigonometric mutation strategy defined by Eq. (20), the value of the parameter τ_Q was set to 0.1 [5]. Population size, as well as, the number of generations per restart were problem dependent. The setup used for these parameters was the same for DE and PSO and their values are reported in Table 5.

Concerning the PSO method, the global variant of the algorithm was considered because it exhibited faster convergence compared to the local variant. In contrast to DE and CMA-ES, the particles were constrained in the box $[-1, 1]^n$, where n stands for the dimension of the problem, in order to avoid possible velocities explosion. In the constriction factor version, the values of the c_1 and c_2 parameters were set to 2.05, while χ was set to 0.729 [3]. An upper bound, $V_{\max}$, on the absolute value of the velocities of the particles was used and set to 1. In the inertia weight version, the inertia weight w was initialized to 1.0 and it gradually declined towards zero for the 75% of the available iterations. Swarm

Table 6
Results for Test Problem 1 with deflection

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	2.97	0.18	2	3	**8830.08**	2048.87	6644.00	17638.00
DE1	2.97	0.18	2	3	21297.67	6006.98	11493.33	44170.00
DE2	2.93	0.25	2	3	25512.56	6892.15	19706.67	49220.00
DE3	2.97	0.18	2	3	22237.22	6631.78	17160.00	52670.00
DE4	**3.00**	0.00	3	3	21990.89	3364.67	17406.67	29346.67
DE5	**3.00**	0.00	3	3	25781.78	3448.10	21273.33	32780.00
DE6	**3.00**	0.00	3	3	23996.67	3220.54	20340.00	30146.67
PSOc	2.97	0.18	2	3	23219.00	5208.07	18020.00	45530.00
PSOi	**3.00**	0.00	3	3	30966.67	2170.56	27860.00	34140.00

Table 7
Results for Test Problem 1 with multistart

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	2.57	0.50	2	3	**3344.69**	731.87	2234.00	4642.50
DE1	2.43	0.50	2	3	4796.33	1124.29	3400.00	7050.00
DE2	2.70	0.47	2	3	8708.00	2551.35	6046.67	13970.00
DE3	2.43	0.50	2	3	6220.22	1556.19	3793.33	10430.00
DE4	**2.77**	0.43	2	3	5085.78	1330.14	3660.00	7970.00
DE5	2.57	0.50	2	3	9749.89	2965.66	5793.33	15820.00
DE6	2.70	0.47	2	3	9478.67	2654.48	6793.33	15290.00
PSOc	2.67	0.48	2	3	6265.56	1849.13	4293.33	10020.00
PSOi	2.47	0.51	2	3	23998.56	5630.68	16986.67	32030.00

size and generations per restart were problem dependent and the setup used in the numerical experiments is summarized in Table 5.

4.4. Presentation of the results

To evaluate the comparative performance of CMA-ES, DE, and PSO, on each test problem, we compared the performance of each algorithm with respect to the number of (different) Nash equilibria detected, as well as, with respect to the mean number of function evaluation required to compute a different Nash equilibrium. Moreover, on each test problem we investigated the performance differences caused by the choice between the deflection and the multistart technique. For each test problem, and for each choice between deflection and restart, 30 numerical experiments were performed for each of the considered algorithms. All results are reported in Tables 6–17. Each table reports the mean, the standard deviation (σ), the minimum (min), and maximum (max) number of different Nash equilibria detected, and the corresponding values for the number of function evaluations required to compute a distinct Nash equilibrium.

Table 8
Results for Test Problem 2 with deflection

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	4.27	0.45	4	5	**9588.79**	1395.51	6927.80	11692.50
DE1	4.73	0.45	4	5	22657.00	3933.48	15432.00	29865.00
DE2	4.30	0.47	4	5	32609.63	4529.02	24388.00	40230.00
DE3	4.63	0.49	4	5	25435.27	4164.88	19844.00	32955.00
DE4	4.33	0.48	4	5	28687.27	4971.97	19860.00	36610.00
DE5	0.87	0.51	0	2	—	—	—	—
DE6	4.47	0.51	4	5	34362.23	5270.18	26620.00	43985.00
PSOc	4.67	0.48	4	5	24504.73	4703.46	19288.00	34285.00
PSOi	**4.90**	0.31	4	5	30276.07	2628.65	28120.00	37910.00

Table 9
Results for Test Problem 2 with multistart

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	3.33	0.76	2	5	**3090.94**	736.69	1991.40	4873.50
DE1	3.37	0.85	2	5	7069.79	2063.60	4072.00	11046.67
DE2	2.73	0.74	1	4	35206.72	16168.87	17480.00	97740.00
DE3	2.83	0.65	2	4	15888.28	6360.15	7766.67	29800.00
DE4	3.30	0.84	2	5	29414.12	13156.15	7408.00	75290.00
DE5	0.97	0.41	0	2	—	—	—	—
DE6	2.87	0.73	2	4	43101.39	14224.91	25795.00	71750.00
PSOc	**3.40**	0.86	2	5	11973.08	3660.57	7348.00	20610.00
PSOi	3.23	0.73	2	5	39025.94	10225.40	24516.00	69670.00

Table 10
Results for Test Problem 3 with deflection

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	3.03	0.89	2	5	**19778.25**	7116.49	9223.80	30626.50
DE1	3.10	0.55	2	4	218134.03	62594.70	118787.50	369475.00
DE2	1.20	0.71	0	3	—	—	—	—
DE3	3.17	0.75	2	4	227278.33	78168.94	137712.50	377850.00
DE4	3.03	0.72	2	5	253272.33	85881.17	108470.00	432625.00
DE5	1.63	0.76	0	3	—	—	—	—
DE6	2.57	0.82	1	4	376246.53	207052.30	160337.50	941800.00
PSOc	3.00	0.69	2	4	255346.25	91762.29	107550.00	422075.00
PSOi	**3.37**	0.72	2	5	255715.97	70688.00	139450.00	451250.00

Table 11
Results for Test Problem 3 with multistart

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	2.43	0.90	1	4	**13076.37**	6503.76	4426.50	28458.00
DE1	2.40	0.81	1	4	57805.69	59371.86	17075.00	342550.00
DE2	1.67	0.71	0	3	—	—	—	—
DE3	2.37	0.72	1	4	66788.47	36564.10	20487.50	189000.00
DE4	**2.77**	0.68	1	4	74955.14	48618.67	24600.00	252400.00
DE5	1.47	0.68	0	3	—	—	—	—
DE6	2.50	0.82	1	4	340859.44	222252.48	156475.00	936450.00
PSOc	2.50	0.63	2	4	78288.75	33132.98	38600.00	157825.00
PSOi	2.30	0.60	1	3	226939.72	78364.01	148750.00	495450.00

Table 12
Results for Test Problem 4 with deflection

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	7.37	0.93	6	9	**4871.85**	1597.16	2091.00	8202.50
DE1	6.70	1.09	4	9	8977.96	3176.34	4145.00	17247.50
DE2	7.17	1.05	5	9	9783.33	2956.87	5838.75	17442.00
DE3	7.27	0.87	6	9	9983.27	2640.40	6037.50	15616.67
DE4	**7.90**	0.76	7	9	7036.02	1814.05	3757.78	11135.71
DE5	6.80	1.13	4	9	14986.52	4051.78	8363.33	30562.50
DE6	7.57	0.90	5	9	9568.29	2586.73	6322.22	18076.00
PSOc	7.03	0.76	5	8	8569.21	1838.82	5630.00	14886.00
PSOi	6.90	0.96	5	8	15026.70	3084.20	10991.25	21912.00

Table 13
Results for Test Problem 4 with multistart

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	2.70	0.65	2	4	**3194.14**	715.61	2052.75	4408.50
DE1	2.73	0.64	2	4	6281.19	3456.75	2752.50	14680.00
DE2	2.77	0.73	2	4	11211.28	3456.54	6526.67	18205.00
DE3	2.53	0.57	2	4	11934.17	4423.44	6950.00	23745.00
DE4	**3.03**	0.85	2	4	11633.22	4497.53	6105.00	26060.00
DE5	2.70	0.70	2	4	37830.64	11964.14	19560.00	65805.00
DE6	2.80	0.81	2	5	16947.42	6346.54	8095.00	32090.00
PSOc	2.93	0.58	2	4	7455.39	1701.89	5442.50	10565.00
PSOi	2.07	0.45	1	4	40458.78	9647.80	18715.00	82550.00

Table 14
Results for Test Problem 5 with deflection

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	10.30	0.60	9	12	**2303.75**	490.27	1556.82	3482.10
DE1	10.40	0.62	10	12	4798.46	1023.89	3511.67	8140.00
DE2	**10.57**	0.63	10	12	9279.43	1045.27	7665.00	11436.00
DE3	10.47	0.57	10	12	5697.87	1306.27	4121.82	10224.00
DE4	9.10	1.06	7	10	21072.97	7679.69	9866.00	36940.00
DE5	0.73	1.53	0	8	—	—	—	—
DE6	10.17	0.38	10	11	17023.22	3706.66	9872.73	22782.00
PSOc	10.53	0.78	9	12	5598.80	844.73	4290.00	7450.00
PSOi	10.50	0.51	10	11	15116.76	1009.87	13783.64	16912.00

Table 15
Results for Test Problem 5 with multistart

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	1.77	0.86	1	4	**7753.11**	3252.50	2863.75	11882.00
DE1	1.30	0.53	1	3	39076.33	11198.85	14360.00	51900.00
DE2	1.70	0.65	1	3	97813.78	41449.85	41866.67	167720.00
DE3	1.33	0.55	1	3	62016.11	18262.06	21433.33	91640.00
DE4	**1.93**	0.78	1	4	149817.17	75240.08	53565.00	310060.00
DE5	0.27	0.52	0	2	—	—	—	—
DE6	1.53	0.82	1	4	203752.00	74901.93	60800.00	289700.00
PSOc	1.40	0.50	1	2	33819.00	10338.13	20350.00	44820.00
PSOi	1.23	0.43	1	2	134998.00	32606.27	76110.00	160380.00

Table 16
Results for Test Problem 6 with deflection

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	13.93	0.58	13	15	**2200.47**	269.03	1671.21	2787.13
DE1	13.90	0.71	12	15	3863.48	594.87	2939.33	5440.00
DE2	13.80	0.48	13	15	5217.48	609.81	4358.57	6937.69
DE3	13.97	0.61	13	15	5835.41	917.29	4361.43	7510.00
DE4	13.70	0.75	12	15	5054.90	595.60	4272.14	6483.08
DE5	11.87	1.14	8	13	9522.17	2106.37	7140.00	18767.50
DE6	13.60	0.67	12	15	5358.69	682.98	4482.14	7304.62
PSOc	14.20	0.89	12	15	3047.69	347.04	2540.00	3834.62
PSOi	**14.33**	0.48	14	15	6815.01	237.90	6276.00	7197.86

Table 17
Results for Test Problem 6 with multistart

Method	Nash equilibria				Function evaluations per equilibrium			
	Mean	σ	Min	Max	Mean	σ	Min	Max
CMA-ES	5.90	0.96	4	8	**2990.22**	531.53	2142.38	4549.00
DE1	6.33	1.12	4	9	6613.78	1598.33	4415.00	12607.50
DE2	6.43	1.28	4	9	11267.65	3036.64	6545.56	18670.00
DE3	5.47	1.38	2	8	14720.26	6609.17	6441.25	41225.00
DE4	**6.97**	1.16	4	9	12314.96	3164.66	8338.89	20575.00
DE5	6.87	1.25	5	9	20609.33	4637.34	14332.22	32652.00
DE6	6.33	1.03	5	8	12927.57	2583.13	9397.50	18496.00
PSOc	5.33	1.12	2	7	6499.35	2123.42	4395.71	15865.00
PSOi	4.60	1.07	2	6	21526.97	7461.19	14948.33	46365.00

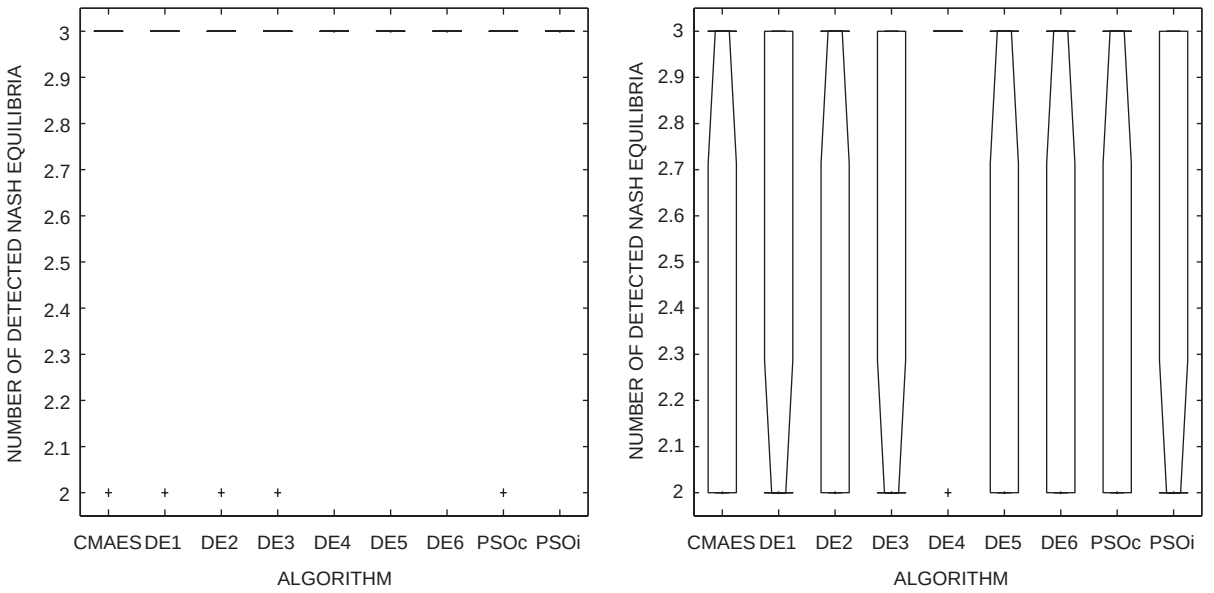

Fig. 1. Nash equilibria detected with deflection (left) and multistart (right) for TP1.

To obtain a clearer image of the statistical properties of the obtained results, we also provide boxplots, in Figs. 1–6 for the detected Nash equilibria, and in Figs. 7–12 for the number of function evaluations. Each box corresponds to an instance of one of the considered algorithms, and it has lines at the lower quartile, median, and upper quartile values. The whiskers, i.e. the lines extending from each end of the box, show the extent covered by the remaining values. Outliers appear beyond the ends of the whiskers, and they are denoted with crosses. Notches represent a confidence interval for the medians for box to box comparison.

As we can see from Table 6, the DE variants 4, 5, and 6, as well as the inertia weight version of PSO (PSOi) are capable of detecting all 3 Nash equilibria of Test Problem 1 in every run. Moreover, no

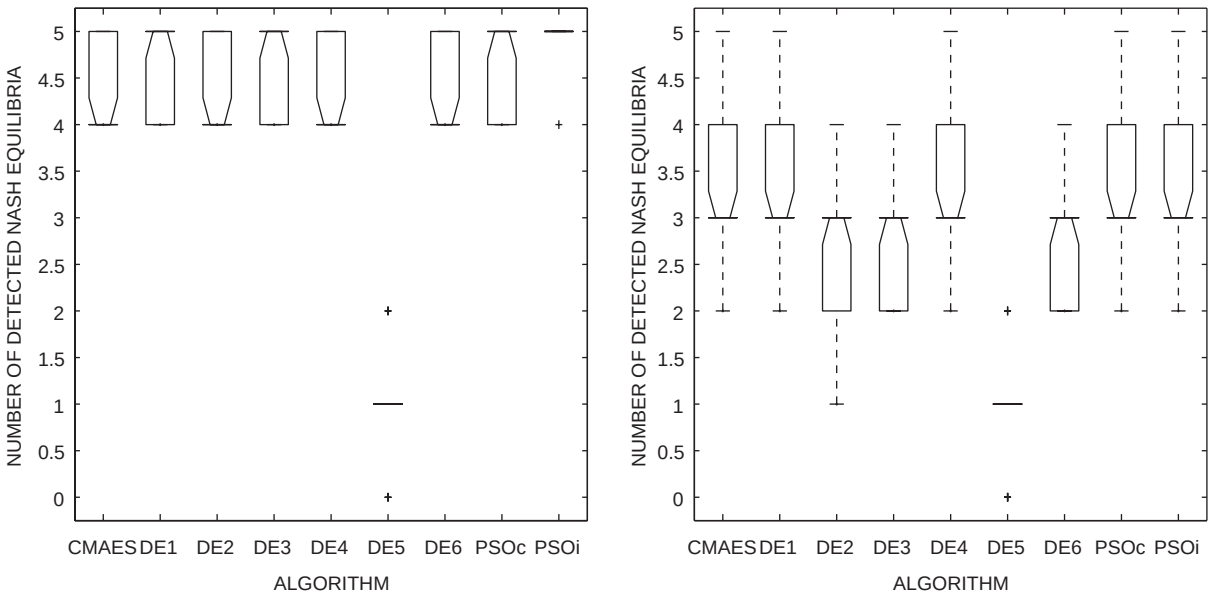

Fig. 2. Nash equilibria detected with deflection (left) and multistart (right) for TP2.

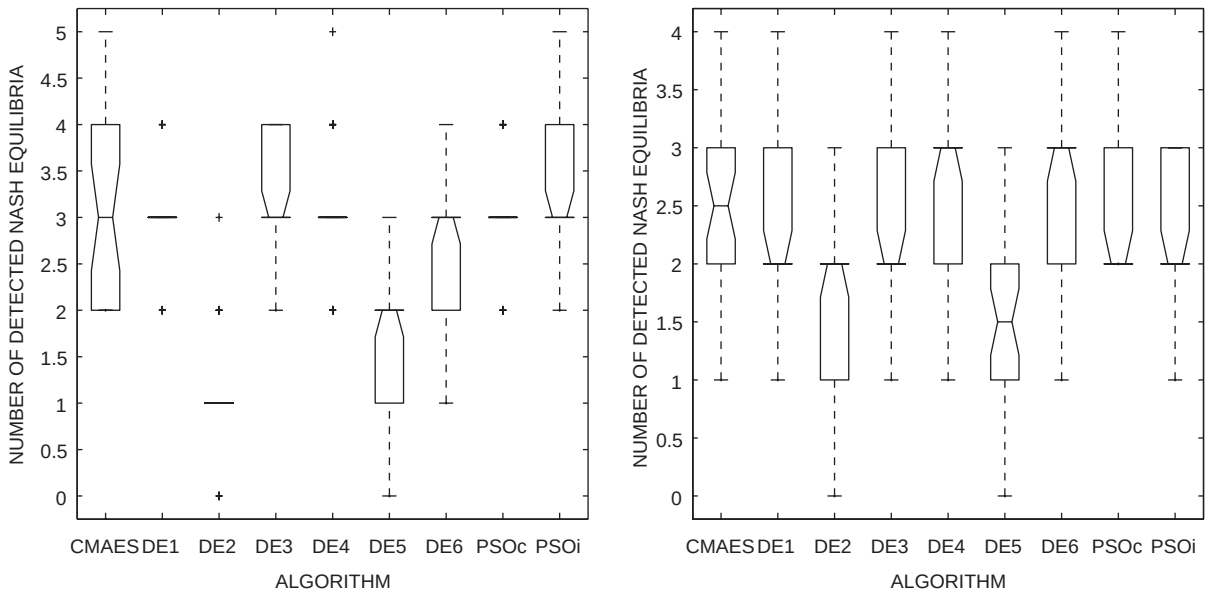

Fig. 3. Nash equilibria detected with deflection (left) and multistart (right) for TP3.

algorithm found less than 2 equilibria in a single run. Performing a Kruskal–Wallis statistical test [8] on the results, the null hypothesis that the median performance of all the algorithms is identical, was not rejected (p-value equal to 0.634). It is important to note at this point that this hypothesis was rejected in all the

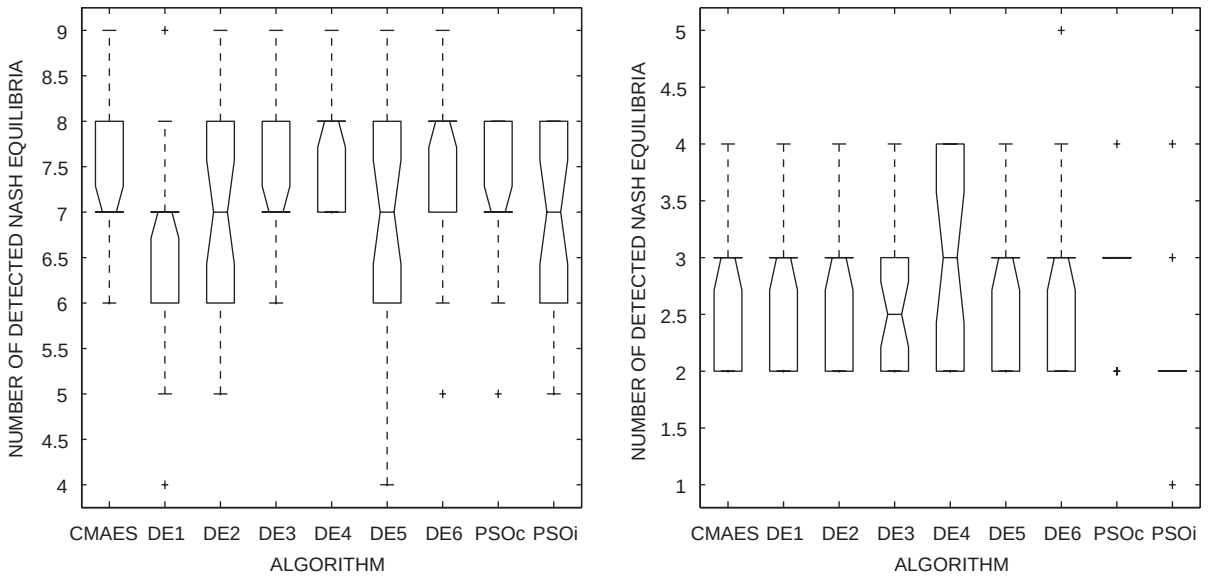

Fig. 4. Nash equilibria detected with deflection (left) and multistart (right) for TP4.

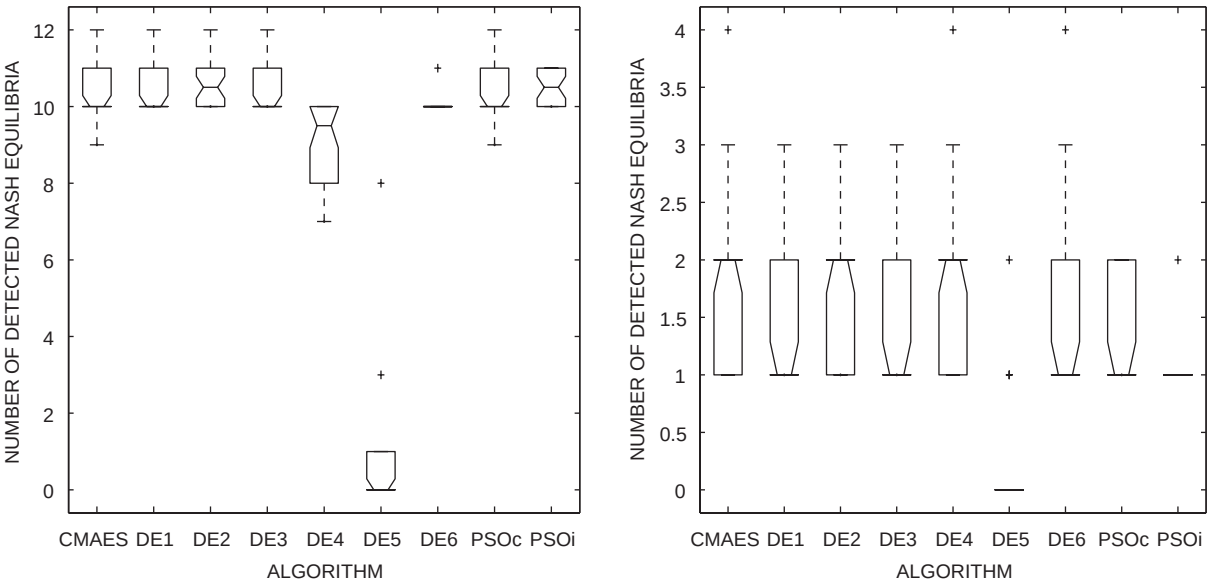

Fig. 5. Nash equilibria detected with deflection (left) and multistart (right) for TP5.

other cases. Comparing the results of Table 6 with those of Table 7 we observe that deflection marginally improves the capability of all the algorithms to compute different minimizers. CMA-ES proved to be the computationally cheapest method. This finding, which is also valid for all the other test problems, was expected since in our experimental setting CMA-ES required the smallest number of individuals compared to DE and PSO.

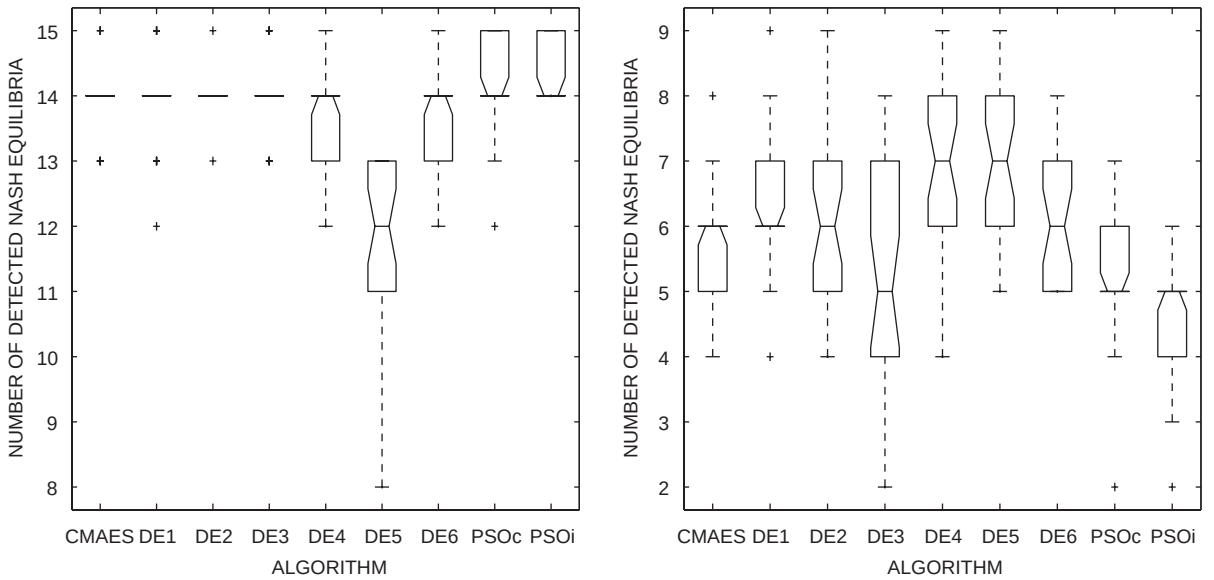

Fig. 6. Nash equilibria detected with deflection (left) and multistart (right) for TP6.

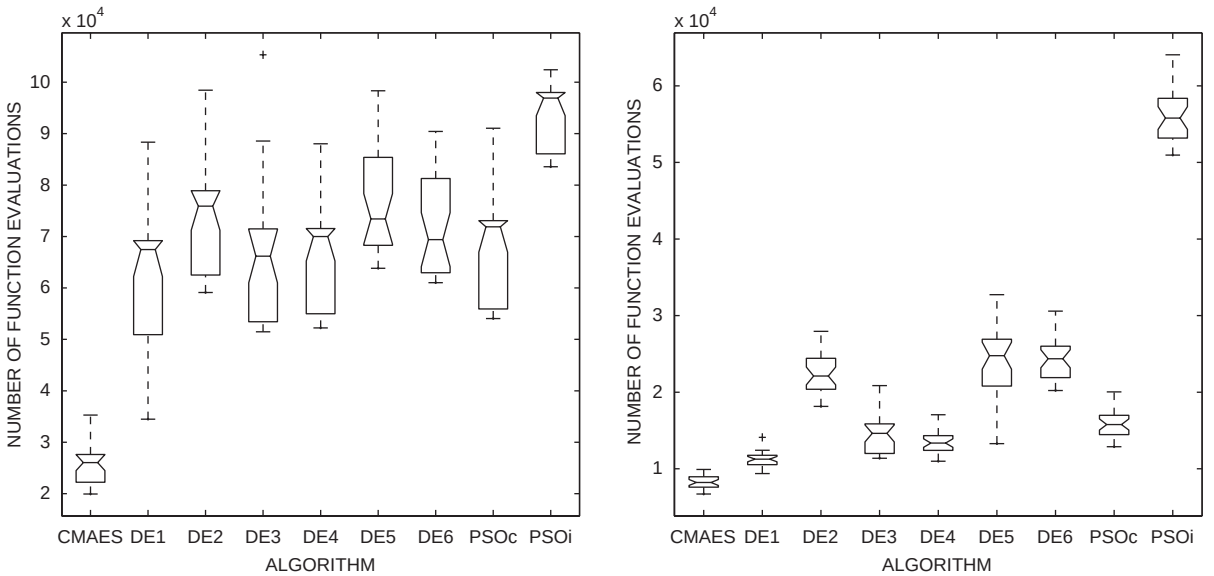

Fig. 7. Function evaluations for deflection (left) and multistart (right) for TP1.

Regarding Test Problem 2, the best performing method with respect to the number of different minimizers detected was PSOi, when deflection was used, and the constriction factor version of PSO (PSOc) when multistart was used. As can be seen from Tables 8 and 9, deflection ensured a superior performance

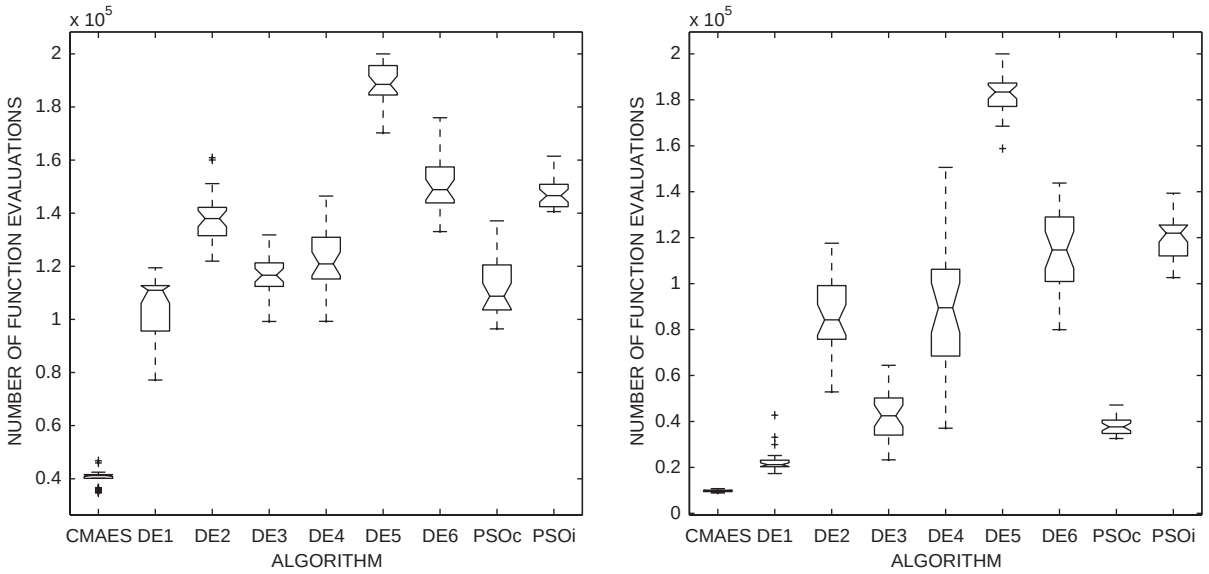

Fig. 8. Function evaluations for deflection (left) and multistart (right) for TP2.

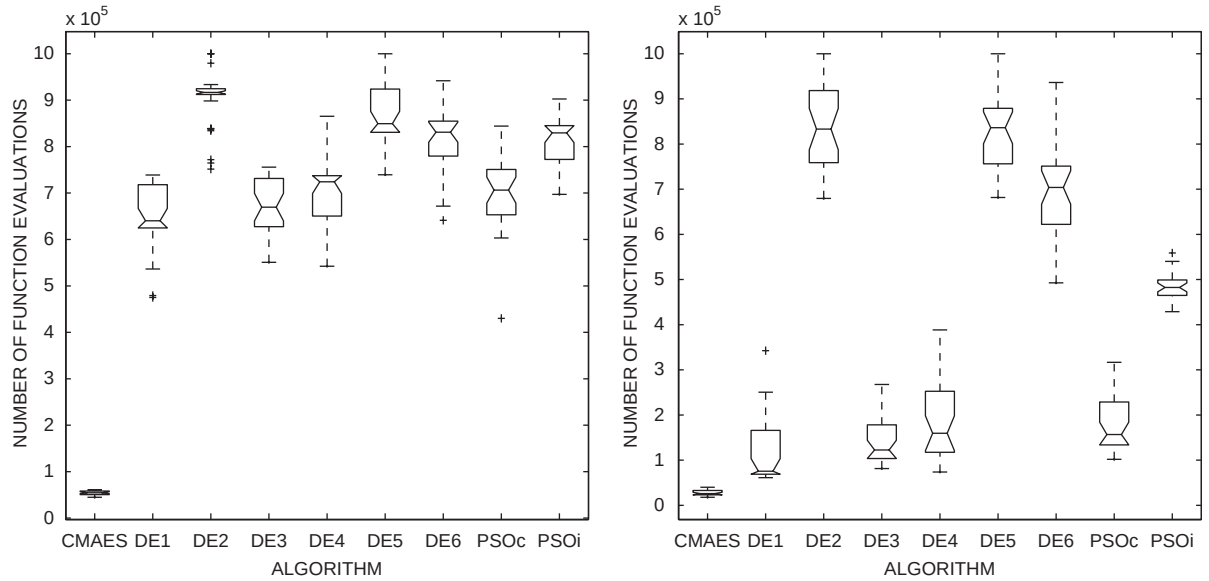

Fig. 9. Function evaluations for deflection (left) and multistart (right) for TP3.

than multistart, since all algorithms (with the exception of DE5) managed to compute at least 4 out of 5 equilibria. This is more clearly shown in Fig. 2. Function evaluations per equilibrium are not reported for DE5 since the algorithm was incapable of detecting even one equilibrium in some occasions.

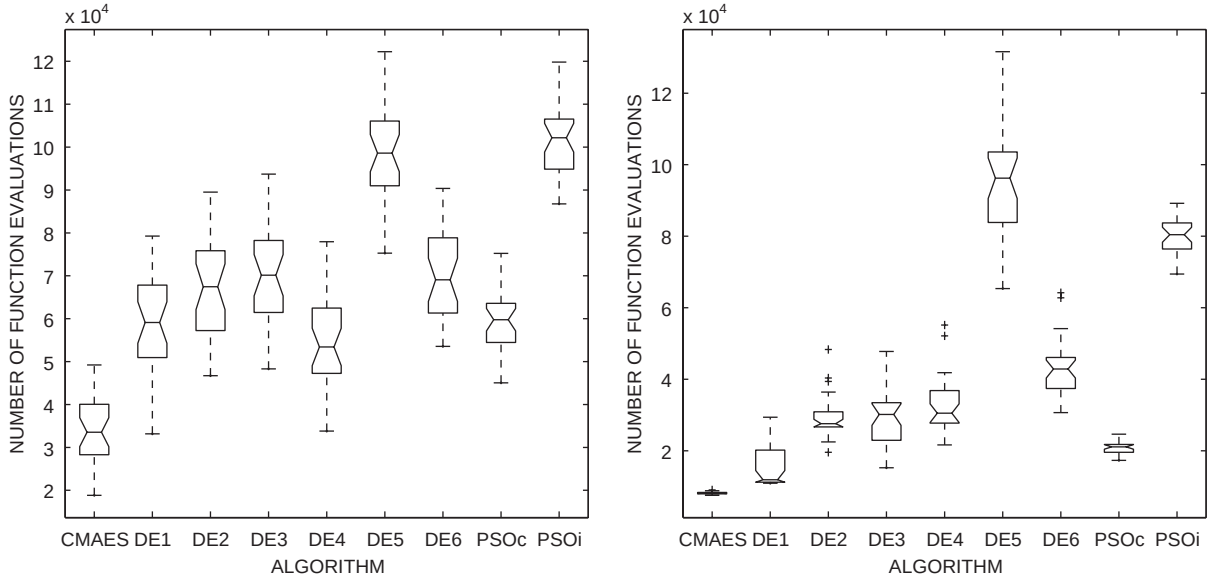

Fig. 10. Function evaluations for deflection (left) and multistart (right) for TP4.

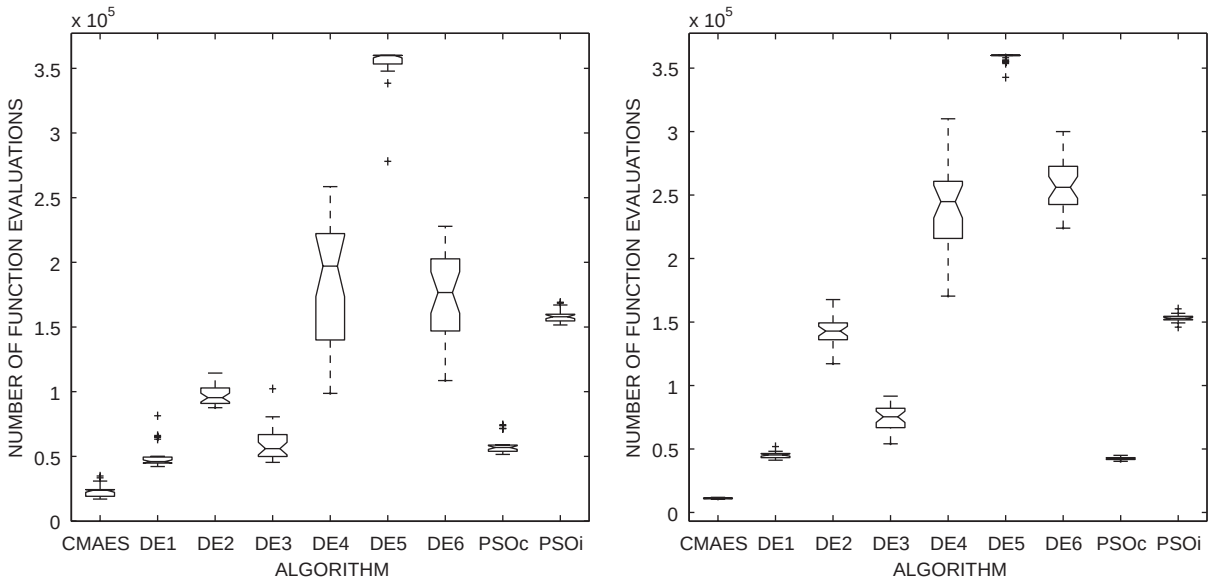

Fig. 11. Function evaluations for deflection (left) and multistart (right) for TP5.

Similar results were obtained for Test Problem 3, as shown in Tables 10 and 11. PSOi proved the best performing method when equipped with deflection. On the other hand, DE4 gave the best results for multistart. A poor performance was exhibited by DE2 and DE5. On Test Problem 4, Tables 12 and 13,

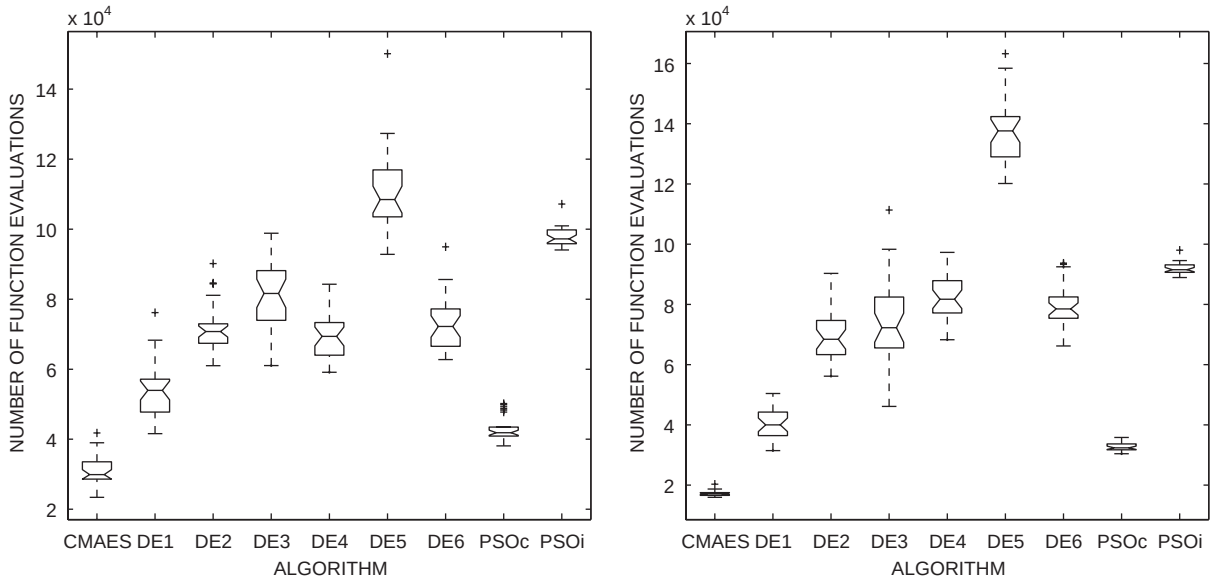

Fig. 12. Function evaluations for deflection (left) and multistart (right) for TP6.

DE4 proved the best performing method when equipped with either deflection or multistart. The same was observed for Test Problems 5 and 6 with multistart, while for the deflection case, DE2 and PSOi performed better, respectively.

Overall, the obtained results suggest that incorporating the deflection technique improves significantly the ability of the considered methods to detect different minimizers, compared to multistart. DE4 exhibits the best overall performance when multistart is used. When equipped with the deflection technique, the best performing methods appear to be DE4, PSOi and PSOc. An interesting point to note is that on this set of problems, the DE variants that exploit the best individual of the population in the mutation strategy (DE1, DE3, and DE4) overall exhibited a superior performance. In the context of computational intelligence methods, this property is known as elitism. Finally, the least computationally expensive method per computed minimizer, was CMA-ES.

5. Conclusions

The concept of Nash equilibrium is central in game theory. In this contribution the effectiveness of three computational intelligence methods, namely CMA-ES, PSO, and DE, was investigated on the task of locating and computing the Nash equilibria of finite strategic games. To employ these methods the global optimization formulation of the problem of computing Nash equilibria was adopted. To detect more than one Nash equilibria in a single run, the multistart and the deflection techniques for the computation of more than one global minimizers of a function, were employed.

In all the test problems, the deflection technique enhanced significantly the performance of the considered methods compared to multistart. This finding becomes more pronounced as the number of equilibria increases. In most cases, the methods that exploit the best individuals of the current population to produce

the population of the next generation, tended to exhibit superior performance. Overall, the two versions of PSO and the DE variants that exploit the best individual in the mutation strategy, exhibited the most robust behavior with respect to the number of different equilibria detected. The CMA-ES was the least computationally expensive method per computed minimizer.

Acknowledgements

We would like to thank the editor and the reviewers for their consideration. We also acknowledge the partial support by the "Pythagoras" research programme awarded by the Greek Ministry of Education and Religious Affairs and the European Union.

References

[1] R. Aumann, S. Hart (Eds.), Handbook of Game Theory, Handbooks in Economics (11), vol. 1, North-Holland, Amsterdam, 1992.
[2] H.-G. Beyer, H.-P. Schwefel, Evolution Strategies: a comprehensive introduction, Natural Comput. 1 (1) (2002) 3–52.
[3] M. Clerc, J. Kennedy, The particle swarm-explosion stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[4] R.C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, in: V.W. Porto, N. Saravanan, D. Waagen, A.E. Eiben (Eds.), Evolutionary Programming, vol. VII, Springer, Berlin, 1998, pp. 611–616.
[5] H.Y. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, J. Global Optim. 27 (2003) 105–129.
[6] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evol. Comput. 11 (1) (2003) 1–18.
[7] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2) (2001) 159–195.
[8] M. Hollander, D.A. Wolfe, Nonparametric Statistical Methods, Wiley, New York, 1973.
[9] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings IEEE International Conference on Neural Networks, vol. IV,IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.
[10] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, Los Altos, CA, 2001.
[11] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, On the alleviation of local minima in backpropagation, Nonlinear Anal. Theory Methods Appl. 30 (7) (1997) 4545–4550.
[12] R.D. McKelvey, A Liapunov function for Nash equilibria, Technical Report, California Institute of Technology, 1991.
[13] R.D. McKelvey, A. McLennan, Computation of equilibria in finite games, in: H.M. Amman, D.A. Kendrick, J. Rust (Eds.), Handbook of Computational Economics, Handbooks in Economics (13), vol. 1, North-Holland, Amsterdam, 1996, pp. 87–142.
[14] R.D. McKelvey, A. McLennan, T. Turocy, Gambit Command Language, California Institute of Technology, 2000.
[15] J.F. Nash, Equilibrium points in *n*-person games, Proc. Natl. Acad. Sci. 36 (1950) 48–49.
[16] J.F. Nash, Noncooperative games, Ann. Math. 54 (1951) 289–295.
[17] M.J. Osborne, A. Rubinstein, A Course in Game Theory, MIT Press, Cambridge, MA, 1994.
[18] K.E. Parsopoulos, M.N. Vrahatis, Initializing the particle swarm optimizer using the nonlinear simplex method, in: A. Grmela, N. Mastorakis (Eds.), Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, WSEAS Press2002, pp. 216–221.
[19] K.E. Parsopoulos, M.N. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Comput. 1 (2–3) (2002) 235–306.
[20] K.E. Parsopoulos, M.N. Vrahatis, Computing periodic orbits of nondifferentiable/discontinuous mappings through particle swarm optimization, in: Proceedings of the IEEE Swarm Intelligence Symposium, Indianapolis, IN, USA, 2003, pp. 34–41.

[21] V.P. Plagianakos, M.N. Vrahatis, Parallel evolutionary training algorithms for "hardware-friendly" neural networks, Natural Comput. 1 (2–3) (2002) 307–322.
[22] I. Rechenberg, Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution, Ph.D. Thesis, Department of Process Engineering, Technical University of Berlin, Germany, 1971.
[23] H.-P. Schwefel, Evolutionsstrategie und numerische optimierung, Ph.D. Thesis, Department of Process Engineering, Technical University of Berlin, Germany, 1975.
[24] H.-P. Schwefel, Numerical Optimization of Computer Models, Wiley, Chichester, 1981.
[25] H.-P. Schwefel, Evolution and Optimum Seeking, Wiley, New York, 1995.
[26] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings IEEE Conference on Evolutionary Computation,IEEE Service Center, Anchorage, AK, 1998, pp. 69–73.
[27] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: V.W. Porto, N. Saravanan, D. Waagen, A.E. Eiben (Eds.), Evolutionary Programming, vol. VII, Springer, Berlin, 1998, pp. 591–600.
[28] B. von Stengel, Computing equilibria for two-person games, in: R. Aumann, S. Hart (Eds.), Handbook of Game Theory, vol. 3, North-Holland, Amsterdam, 2002, pp. 1723–1759 (Chapter 45).
[29] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[30] A. Törn, A program for global optimization, multistart with clustering (MSC), in: Proceedings of Euro IFIP,1979, pp. 427–434.
[31] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inform. Process. Lett. 85 (2003) 317–325.