Struct Multidisc Optim (2013) 47:367–388
DOI 10.1007/s00158-012-0851-z

RESEARCH PAPER

A mixed-discrete Particle Swarm Optimization algorithm with explicit diversity-preservation

Souma Chowdhury · Weiyang Tong · Achille Messac · Jie Zhang

Received: 28 October 2011 / Revised: 4 October 2012 / Accepted: 15 October 2012 / Published online: 13 December 2012

Abstract Engineering design problems often involve nonlinear criterion functions, including inequality and equality constraints, and a mixture of discrete and continuous design variables. Optimization approaches entail substantial challenges when solving such an all-inclusive design problem. In this paper, a modification of the Particle Swarm Optimization (PSO) algorithm is presented, which can adequately address system constraints while dealing with mixed-discrete variables. Continuous search (particle motion), as in conventional PSO, is implemented as the primary search strategy; subsequently, the discrete variables are updated using a deterministic *nearest-feasible-vertex* criterion. This approach is expected to alleviate the undesirable difference in the rates of evolution of discrete and continuous variables. The premature stagnation of candidate solutions (particles) due to loss of diversity is known to be one of the primary drawbacks of the basic PSO dynamics. To address this issue in high dimensional design problems, a new adaptive diversity-preservation technique is developed. This technique characterizes the population diversity at each iteration. The estimated diversity measure is then used to apply (i) a dynamic repulsion away from the best global solution in the case of continuous variables, and (ii) a stochastic update of the discrete variables. For performance validation, the Mixed-Discrete PSO algorithm is applied to a wide variety of standard test problems: (i) a set of 9 unconstrained problems, and (ii) a comprehensive set of 98 Mixed-Integer Nonlinear Programming (MINLP) problems. We also explore the applicability of this algorithm to a large scale engineering design problem—-wind farm layout optimization.

Keywords Constraint · Discrete variable · Mixed-integer nonlinear programming (MINLP) · Particle Swarm Optimization · Population diversity · Wind farm layout

1 Introduction

Particle Swarm Optimization (PSO) is a stochastic optimization algorithm that imitates the dynamics of social behavior observed in nature. This algorithm was introduced by an Electrical Engineer, Russel C. Eberhart, and a Social Psychologist, James Kennedy (1995). The underlying philosophy of PSO and swarm intelligence can be found in the book by Kennedy et al. (2001). PSO has emerged over the years to be one of the most popular population-based heuristic optimization approaches. Several variations of PSO have been reported in the literature, and applied to diverse optimization problems in engineering, basic sciences and finance (Banks et al. 2008). The modifications of the PSO algorithm presented in this paper are inspired by

Parts of this manuscript have been presented at the 53rd AIAA Structures, Structural Dynamics and Materials Conference, in April, 2012, at Honolulu, Hawaii - Paper Number: AIAA 2012-1678.

S. Chowdhury · J. Zhang
Multidisciplinary Design and Optimization Laboratory, Department of Mechanical, Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

W. Tong
Multidisciplinary Design and Optimization Laboratory, Department of Mechanical and Aerospace Engineering, Syracuse University, Syracuse, NY 13244, USA

A. Messac (✉)
Department of Mechanical and Aerospace Engineering, Syracuse University, Syracuse, NY 13244, USA
e-mail: messac@syr.edu

the authors' research in product family design (Chowdhury et al. 2010a, 2011) and wind farm optimization (Chowdhury et al. 2010b, 2012). Both of these optimization problems (defined as single objective) involve complex multimodal criterion functions and a high dimensional system of mixed-discrete design variables. These problems are challenging, and generally require a large number of system-model evaluations.

In the case of *constrained single-objective optimization* problems, population-based algorithms (e.g., evolutionary and swarm-based optimization methods) often suffer from premature stagnation (Banks et al. 2008). This undesirable property can be attributed to an excessive and mostly unopposed pressure of exploration or evolution. The simultaneous presence of continuous and discrete design variables that may experience differing rates of evolution further complicates the optimization scenario. In this paper, a new method is developed to both characterize and adaptively infuse diversity into the population of candidate solutions. This method is an evolution from earlier diversity-preservation methods reported in the PSO literature, which are later discussed in Section 1.2. The PSO algorithm presented in this paper can address a mixture of discrete and continuous design variables. Two distinct yet mutually coherent approaches are developed to address the diversity-preservation issues for discrete and continuous variables.

In the following two sections (Sections 1.1 and 1.2), we provide brief surveys of *Mixed-Discrete Nonlinear Optimization* (MDNLO) methodologies and the major variations of the Particle Swarm Optimization algorithm reported in the literature. Section 1.3 introduces the basic principles and objectives of the *Mixed-Discrete Particle Swarm Optimization* (MDPSO) algorithm developed in this paper. Sections 2 and 3 describe the development of the MDPSO algorithm and the generalized diversity characterization/preservation technique, respectively. Results and subsequent discussions regarding the application of MDPSO to various standard test problems, and a real-life engineering problem, a wind farm layout optimization, are given in Section 4.

1.1 Existing mixed-discrete optimization approaches

A significant amount of research has been done in developing algorithms for solving *Mixed-Integer Non-Linear Programming* (MINLP) problems. Most of these algorithms are gradient-based search techniques. Three major categories of gradient-based algorithms are (i) the branch and bound, (ii) the cutting plane, and (iii) the outer approximation algorithms. A list of these algorithms, related discussion, and bibliography can be found in the websites of the MINLP World (2010) and the CMU-IBM Cyber-Infrastructure for MINLP (2010). These algorithms possess attractive numerical properties, namely (i) fast convergence, (ii) proof of optima, and (iii) an intrinsic ability to deal with constraints. However, gradient-based algorithms do not readily apply to the broad scope of engineering design problems that may involve highly nonlinear, non-smooth and multimodal criterion functions.

Among population-based optimization methods, binary Genetic Algorithms (GAs) (Goldberg 1989; Deb 2009) have been reported to be effective for discrete optimization. Binary GAs convert the design variables into binary strings. This process leads to an approximate discrete representation of the continuous variables. A population of candidate solutions, each represented by a binary string, evolve over generations generally through four stages: (i) fitness assignment, (ii) selection, (iii) crossover, and (iv) mutation. One of the most popular binary GAs is the bin-NSGA-II developed by Deb et al. (2002). Genetic algorithms have been successfully implemented on MINLP problems, such as batch plant design (Ponsich et al. 2007, 2008). Another class of discrete optimization algorithms, which belong to Ant Colony Optimization (ACO), have also been reported in the literature (Corne et al. 1999; Bonabeau et al. 1999). Applications of ACO-based algorithms to discrete optimization problems include vehicle routing, sequential ordering, and graph coloring. There exists in the literature a handful of variations of the PSO algorithm that can address discrete and/or integer variables. A summary of these variations of PSO is discussed in the following section.

1.2 Existing Particle Swarm Optimization algorithms

A comprehensive review of the background and the development of Particle Swarm Optimization-based algorithms (until 2007) can be found in a chapter by Banks et al. (2007). An extensive follow up review of the various attributes of PSO, and the applicability of PSO to different classes of optimization problems, such as unconstrained/constrained, combinatorial, and multicriteria optimization, can be found in a book chapter by Banks et al. (2008). In this section, we provide a brief survey of reported variations of PSO that address the following critical optimization attributes: (i) mixed-discrete variables, (ii) population diversity preservation, and (iii) constraint handling.

A balance between exploration, exploitation, and population-diversity in PSO requires appropriate quantification of the PSO coefficients, or what is more popularly termed *parameter selection*. One of the earliest strategies to balance exploration and exploitation was the introduction of the inertia weight (Banks et al. 2007). Eberhart (1998) investigated the influences of the inertia weight and the maximum velocity on the algorithm performance.

Using numerical experiments, they proposed particular values (and/or range of values) for the inertia weight and the maximum velocity, and also suggested the application of time varying inertia weight to further improve the algorithm performance. Trelea (2003) used standard results from dynamic systems theory to provide graphical parameter selection guidelines. The applications of control theory by Zhang et al. (2009), and chaotic number generation by Alatas et al. (2009) are among the recently proposed methods used to establish parameter selection guidelines (for PSO).

Several variations of the PSO algorithm that can solve combinatorial optimization problems have been reported in the literature. Kennedy and Eberhart (1997) presented one of the earliest modification of PSO to address binary variables. They defined the trajectories of the binary variables in terms of the change in the probability that a value of one or zero will be taken. Tasgetiren et al. (2007) used construction/destruction operators to perturb the discrete component of the variable vector of a particle in solving a Traveling Salesman problem. A similar combinatorial-PSO concept was also developed and used by Jarboui et al. (2008) for resource-constrained project scheduling. These variations of the PSO algorithm provide efficient and robust performances, typically for combinatorial optimization problems that are similar to the corresponding reported applications. A majority of these methods do not readily apply to the broad scope of mixed-discrete optimization that involves problems with: (i) *integers and/or real-valued discrete variables*, (ii) *non-uniformly spaced discrete variable values* (e.g., $x \in [1, 3, 100, 1000, \ldots]$) and (iii) *widely different sizes of the "set of feasible values" for the discrete variables* (e.g., $x_1 \in [0, 1]$ and $x_2 \in [1, 2, \ldots, 1000]$).

Kitayama et al. (2006) developed a more generalized approach to address discrete variables using a penalty function—-discrete variables are treated as continuous variables by penalizing at the intervals. However, the additional multimodal constraint in the penalty function-based approach may undesirably increase the complexity of the design problem. Singh et al. (2010) presented an interesting approach to address discrete variables, which manipulates the random operators in the particle-velocity update step. This approach can be very helpful in maintaining consistency in the rates of evolution of the continuous and the discrete variables. The needed *stochastic* and *mutually independent* attributes of the random operators that regulate the PSO dynamics are restricted in this approach.

Preservation of the population diversity to avoid premature convergence has been a long-standing challenge for PSO. Rapid swarm convergence, which is one of the key advantages of PSO over other population-based algorithms, can however lead to stagnation of particles in a small suboptimal region. Efficient and time-variant parameter selection has been traditionally used as an implicit method to avoid particle stagnation, thereby preserving population diversity. Over the years, the use of explicit diversity preservation techniques have proved to be more effective (Kennedy and Eberhart 1995). Krink et al. (2002) introduced a collision-avoidance technique to mitigate premature convergence. Particles coming within a defined vicinity of each other were allowed to bounce off; bouncing back along the old velocity vector (U-turn approach) was found to be most effective. Blackwell and Bentley (2002) also developed a diversity preserving swarm based on a similar collision-avoidance concept. The collision avoidance schemes however require an intuitive specification of the threshold radius.

A more globally applicable approach was developed by Riget and Vesterstrom (2002), where the usual attraction phase was replaced by a repulsion phase, when the entire population diversity fell below a predefined threshold. In this case, the usual PSO location update formula is applied with the direction reversed. A metric similar to the standard deviation of the particle locations was used as the measure of diversity. This measure, however, does not readily account for the combined effects of the distribution of the particles and the overall spread of the particles in the variable space. In other words, with their method (Riget and Vesterstrom 2002), infrequent extreme deviations (i.e., a higher kurtosis such as $[0, 0, 0, 0, 10, -10]$) may yield the same measure of diversity as frequent moderate deviations (e.g., $[5, 6, 7, -5, -6, -7]$), which is misleading. Other interesting methodologies to address population diversity include: (i) introduction of a predatory particle (Silva et al. 2002), and (ii) introduction of the concept of negative entropy from thermodynamics (Xie and Yang 2002). *Nevertheless, the consideration of population diversity in a mixed-discrete/combinatorial optimization scenario (in PSO) has rarely been reported in the literature.*

The basic dynamics of PSO does not account for system constraints. Several variations of the PSO algorithm that incorporate a constraint handling capability have been proposed: (i) a straight-forward method of considering only feasible particles for the best global and the best local solutions (Hu and Eberhart 2002), (ii) the use of conventional dynamic penalty functions (Parsopoulos and Vrahatis 2002), (iii) an effective bi-objective approach where the net constraint serves as the the second objective (Venter and Haftka 2009), and (iv) the use of the efficient constrained non-dominance principles (Zavala et al. 2005). In this paper, we implement the rules of constrained non-dominance introduced by Deb (2009). *Interestingly, the constrained non-dominance principle can be perceived as an aspect of natural swarm intelligence: communication of information from particle to particle regarding whether they are beyond the feasible domain boundaries, and/or how far beyond they are.*

1.3 Mixed-Discrete Particle Swarm Optimization: principles and objectives

This paper presents fundamental modifications to the original dynamics of PSO, with the aim to solve *highly constrained single-objective mixed-discrete optimization* problems. The development of this Mixed-Discrete PSO (MDPSO) is driven by the following specific objectives:

i. Develop an approximation technique that can address mixed-discrete design variables through continuous optimization;
ii. Include a constraint handling technique to deal with both equality and inequality constraints; and
iii. Formulate an explicit diversity preservation technique to avoid the stagnation of particles.

Efficient diversity preservation (the third objective) provides an environment conducive to accomplishing the first and the second objectives. Hence, the third objective is considered to be the primary contribution of this paper. A method is formulated to characterize the existing diversity in the population and adjust the diversity parameter(s)/coefficient(s) at every iteration. *This approach provides a generalized adaptive regulation of the population diversity, which can be implemented in a majority of population-based optimization algorithms and is not restricted to PSO*. For example, the concerned diversity parameter can be (i) the *mutation probability* in genetic algorithms (Deb 2009), or (ii) the *time-varying acceleration coefficients* (TVAC) in PSO (Ratnaweera et al. 2004) or (iii) the *window-size of the hypercube operator* in Predator–Prey algorithms (Chowdhury and Dulikravich 2010), or (iv) the *random selection rate* in Ant Colony Optimization (Nakamichi and Arita 2004).

A majority of the existing Mixed-Discrete PSO algorithms are hindered by the effects of differing rates of evolution of the continuous and discrete design variables. To avoid this undesirable scenario, continuous optimization is applied as the primary search strategy for all variables, whether they are continuous or discrete. After the particles have moved to their new locations, the discrete component of the design vector for each particle is approximated to the nearest feasible discrete domain location. In this case, nearness is determined using the Euclidian distance in the discrete variable space. As a result, although the variables evolve through continuous search dynamics, system-function evaluations are performed only at the allowed discrete locations. This approach is partly similar to the strategy presented by Laskari et al. (2002). A schematic of the proposed mixed-discrete optimization approach for a single particle at a particular iteration is shown in Fig. 1. The term "*feasible discrete space location*" as discussed in this

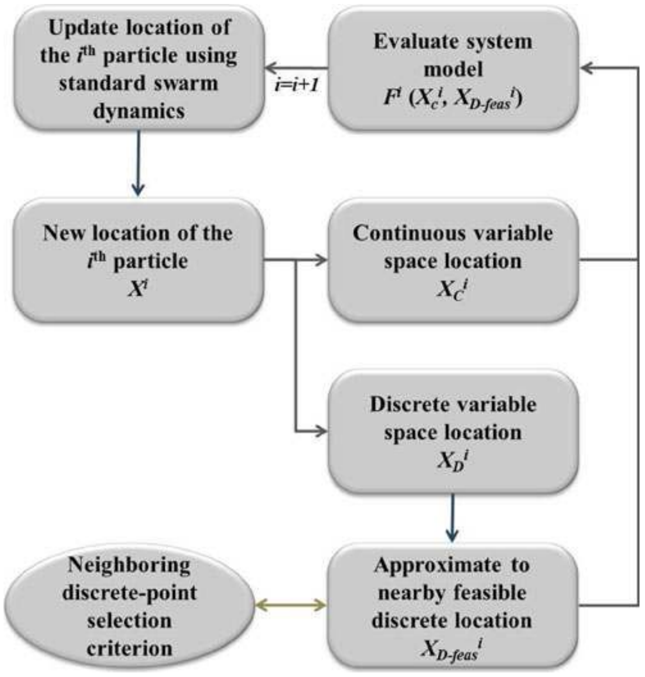

Update location of the i^{th} particle using standard swarm dynamics
Evaluate system model $F^i(X_c^i, X_{D\text{-}feas}^i)$
$i=i+1$
New location of the i^{th} particle X^i
Continuous variable space location X_C^i
Discrete variable space location X_D^i
Neighboring discrete-point selection criterion
Approximate to nearby feasible discrete location $X_{D\text{-}feas}^i$

Fig. 1 Flowchart for the mixed-discrete optimization approach in MDPSO

Section (and as appears in Fig. 1) pertains to the feasibility with respect to the constraints imposed by the discreteness of the variable space, and not to the system constraints.

Constraint handling in MDPSO is performed using the principle of constrained non-dominance that was introduced by Deb et al. (2002). This method has been successfully implemented in the Non-dominated Sorting Genetic Algorithm-II (Deb et al. 2002), Modified Predator–Prey algorithm (Chowdhury and Dulikravich 2010), and other standard evolutionary algorithms. The MDPSO algorithm involves a set of coefficients that regulate the inertia, the personal behavior, the social behavior, and the diversity preserving behavior of the particles. Parameter selection in PSO is far from trivial, as discussed in the previous section. However, detailed analysis of the selection of PSO parameters, and the ensuing numerical behavior of the particle dynamics are not within the scope of this paper. In this paper, we specifically intend to provide:

i. the detailed formulation of the Mixed-Discrete PSO algorithm,
ii. the underlying hypothesis supporting the proposed modifications, and
iii. the performance of this modified algorithm on a wide variety of test cases.

Over the past decade, a substantial amount of interesting research in PSO has been reported in the literature. Effective characteristic modifications can therefore be adopted from the existing variations of the algorithm to further advance the performance of the MDPSO algorithm. For validation purposes, the MDPSO algorithm is applied to (i) a set of standard unconstrained nonlinear optimization problems (Pohlheim 2010; Miele and Cantrell 1969), and (ii) a comprehensive set of MINLP problems (Schittkowski 2009).

2 Development of Mixed-Discrete Particle Swarm Optimization (MDPSO)

2.1 Basic swarm dynamics

A general mixed-discrete single objective constrained minimization problem involving m discrete variables and a total of n design variables can be expressed as

$$
\begin{aligned}
&\text{Min } f(X)\\
&\text{subject to}\\
&\qquad g_j(X) \le 0, \quad j = 1, 2, ..., p\\
&\qquad h_k(X) = 0, \quad k = 1, 2, ..., q\\
&\text{where}\\
&\qquad X = \begin{bmatrix} x_1 & x_2 & \ldots & x_m & x_{m+1} & \ldots & x_n \end{bmatrix}
\end{aligned}
\tag{1}
$$

where p and q are the number of inequality and equality constraints, respectively. In (1), X is the design variable vector, where the first m variables are discrete and the next $n-m$ variables are continuous. To solve this optimization problem, the PSO algorithm is initialized with N random particles. To this end, the Sobol's quasirandom sequence generator (Sobol 1976) is applied. Sobol sequences use a base of two to form successively finer uniform partitions of the unit interval, then reorder the coordinates in each dimension. The location of each particle in the swarm is updated using a velocity vector at each iteration; the velocity vector of a particle is variable, and is itself updated at every iteration. In the MDPSO algorithm, the velocity vector update formula is redefined to allow for an explicit diversity preservation term.

The modified dynamics of the particle motion can be represented as

$$
\begin{aligned}
X_i^{t+1} &= X_i^t + V_i^{t+1},\\
V_i^{t+1} &= \alpha V_i^t + \beta_l r_1 \left(P_i - X_i^t\right) + \beta_g r_2 \left(P_g - X_i^t\right)\\
&\quad + \gamma_c r_3 \hat{V}_i^t
\end{aligned}
\tag{2}
$$

where,

- X_i^t and X_i^{t+1} are the locations of the i^{th} particle at the t^{th} and the $(t+1)^{\text{th}}$ iterations, respectively;
- V_i^t and V_i^{t+1} are the velocity vectors of the i^{th} particle at the t^{th} and the $(t+1)^{\text{th}}$ iterations, respectively;
- r_1, r_2 and r_3 are real random numbers between 0 and 1;
- P_i is the best candidate solution found for the i^{th} particle;
- P_g is the best candidate solution for the entire population (also known as the *current best global solution*);
- α, β_l and β_g are the user defined coefficients that respectively control the inertial, the exploitive, and the explorative attributes of the particle motion;
- γ_c is the diversity preservation coefficient for continuous design variables; and
- the last term $\gamma_c r_3 \hat{V}_i^t$ in the velocity update expression is the diversity preservation term, in which the parameter $\hat{V}_i^t$ is a diverging velocity vector.

The conventional particle dynamics in PSO encourages the particles to congregate, often leading to premature convergence. The purpose of the diverging velocity vector, $\hat{V}_i^t$, is to introduce a direction of motion (in each particle) that opposes such premature congregation of particles. Two different choices for the diverging velocity vector, $\hat{V}_i^t$, are explored in this paper: (i) the vector connecting the mean of the population of solutions to the concerned particle, and (ii) the vector connecting the *current best global solution* to the concerned particle, which is $\hat{V}_i^t = X_i^t - P_g$. Numerical experiments showed that the vector $X_i^t - P_g$ that repels the particle from the best global solutions is more suitable for diversity preservation. A representative illustration of the velocity vectors guiding the motion of a particle (at every iteration) according to the modified velocity update expression is shown in Fig. 2. In this figure, the dotted vector (pointed up and right) represents the original new velocity vector of particle-j, i.e., when the diversity term is not included. It is seen from Fig. 2 that the presence of the diversity vector (pointed down and left) reduces the tendency of the particles to congregate towards the global best particle.

The determination of the diversity preservation coefficient (γ_c) is discussed in Section 3. The best global (P_g) and the best local (P_i) solutions are updated at every iteration using the solution comparison principle. This solution

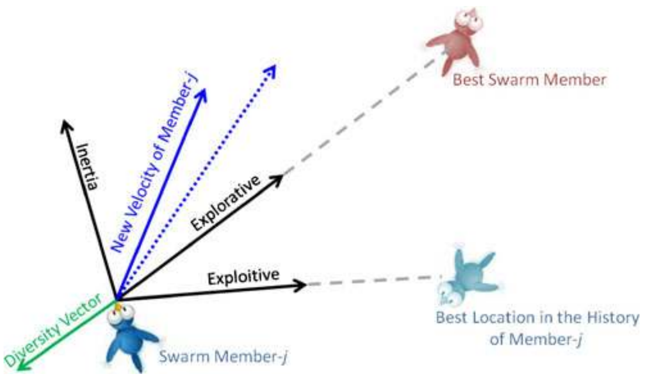


Fig. 2 Modified particle dynamics as given by (2)

comparison principle is based on the values of the objective functions and the constraint functions of the candidate solutions being compared. This principle is discussed in Section 2.3. The continuous update process (2) is applied to *all* the design variables of a particle, irrespective of whether they are continuous or discrete. This approach promotes coherent rates of evolution of the continuous and discrete variables. Following the continuous update process, the discrete component of the design vector is updated to nearby feasible discrete locations. As in the previous Section, feasibility in this case pertains to the constraints imposed by the discreteness of the variable space, and not to the system constraints.

2.2 Updating discrete design variables

In a mixed-discrete optimization scenario, the design space can be divided into a continuous domain and a discrete domain, which correspond to the continuous and the discrete components of the design variable vector, respectively. Following a continuous search PSO step (2), the location of a particle in the discrete domain is defined by a local hypercube that is expressed as

$$H_d = \left\{ \left(x_1^L, x_1^U\right), \ \left(x_2^L, x_2^U\right), \ \ldots, \ \left(x_m^L, x_m^U\right) \right\},$$
$$x_i^L \leq x_i \leq x_i^U, \quad \forall\, i = 1, 2, \ldots, m \tag{3}$$

In (3), m is the number of discrete design variables, and x_i denotes the current location of the candidate solution in the discrete domain. The parameters x_i^L and x_i^U represent two consecutive feasible values of the i^{th} discrete variable that define the boundaries of the local hypercube. The total number of vertices in the hypercube is equal to 2^m.

The values, x_i^L and x_i^U, can be obtained from the discrete vectors that need to be specified a priori for each discrete design variable. A relatively straight-forward criterion, called the Nearest Vertex Approach (NVA), is developed to approximate the current discrete-domain location of the candidate solution to one of the vertices of its local hypercube, H_d (3). The NVA approximates the discrete-domain location to the nearest vertex of the local hypercube (H_d), on the basis of the Euclidean distance. This approximation is represented by

$$\tilde{X} = [\ \tilde{x}_1 \quad \tilde{x}_2 \quad \cdots \quad \tilde{x}_m\ ],$$

where

$$\tilde{x}_i = \begin{cases} x_i^L\,, & \text{if } \left|x_i - x_i^L\right| \leq \left|x_i - x_i^U\right| \\ x_i^U\,, & \text{otherwise} \end{cases}$$
$$\forall\, i = 1, 2, \ldots, m \tag{4}$$

In (4), $\tilde{X}$ represents the approximated discrete-domain location based on the NVA.

Another approach to approximate discrete domain locations was also explored—this approach is called the Shortest Normal Approach (SNA) (Chowdhury et al. 2010a). The SNA approximates the discrete domain location of a particle to the local hypercube vertex that has the *shortest normal distance* from the latest velocity vector of the particle. Numerical experiments showed that the NVA is significantly less expensive and more reliable than the SNA; hence, NVA is used for the application of MDPSO to the test problems in this paper. An illustration of the NVA and the SNA for a 2-D discrete domain is shown in Fig. 3.

This deterministic approximation seeks to retain the search characteristics of the continuous PSO dynamics, while ensuring that the system-model is evaluated only at the allowed discrete domain locations. Such an approximation strategy can be readily implemented in other non-gradient based continuous optimization algorithms as a post process to the usual continuous search step at every iteration.

2.3 Solution comparison and constraint handling

Solution comparison is essential in PSO at every iteration, to determine and update the best global solution in the population and the best local solution for each particle. The principle of constrained non-domination (Deb et al. 2002) is used to compare solutions. According to this principle, candidate solution-i is said to dominate candidate solution-j if and only if one of the following scenarios occur:

I. Solution-i is feasible and solution-j is infeasible or,
II. Both solutions are infeasible and solution-i has a smaller net constraint violation than solution-j or,

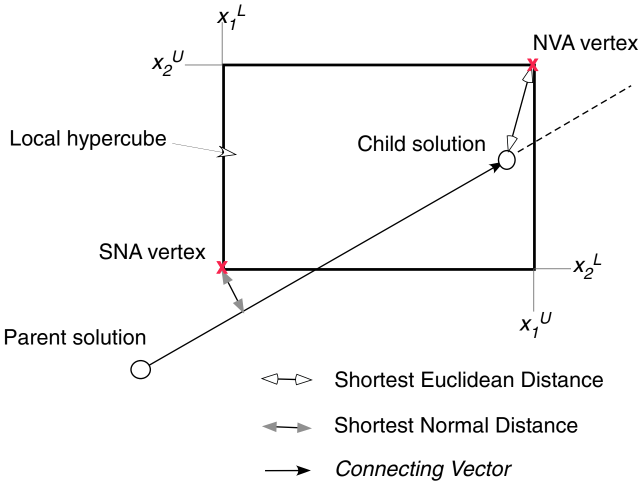


Fig. 3 Illustration of the NVA and the SNA approximation

III. Both solutions are feasible; in addition, solution-i is not worse than solution-j in any objective, and solution-i is better than solution-j in at least one objective.

In the case of a multi-objective problem, it is possible that none of the above scenarios apply, which implies that the solutions are non-dominated with respect to each other.

The net constraint violation $f_c(X)$ is determined by

$$f_c(X) = \sum_{j=1}^{p} \max\left(\bar{g_j}, 0\right) + \sum_{k=1}^{q} \max\left(\bar{h_k} - \epsilon, 0\right) \quad (5)$$

where $\bar{g_j}$ and $\bar{h_k}$ represent the normalized values of the jth inequality constraint and kth equality constraint, respectively. In (5), ϵ represents the tolerance specified to relax each equality constraint; a tolerance value of 1.0e−06 is used for the case studies in this paper. The solution comparison approach in MDPSO favors feasibility over the objective function value. This approach has a tendency to drive solutions towards and into the feasible region during the initial iterations of the algorithm (Chowdhury and Dulikravich 2010; Chowdhury et al. 2009). Throughout this initial phase, dominance scenarios I and II are prominently active. When a majority of the particles have moved into the feasible space, scenario III takes over; solution comparisons are then progressively determined by the magnitude of the objective function.

In the case of highly constrained single-objective problems, this solution comparison approach, together with the intrinsic swarm dynamics, can lead to an appreciable loss in diversity. This undesirable phenomenon occurs primarily during the feasibility-seeking process of optimization. To counter this undesirable characteristic of the particle motion in the MDPSO, the explicit diversity preservation term, $\gamma_c r_3 \hat{V}_i^t$ (refer (2)), is added to the velocity vector, as introduced in Section 2.1.

3 Diversity preservation

The first step in diversity preservation is to characterize and quantify the existing population diversity with respect to the design variable space. A consistent measure of diversity should simultaneously capture the *overall spread* and the *distribution* of the particles in the population. Deb et al. (2002) used a performance metric to measure the spread of solutions along the computed Pareto front in the objective space. A similar metric, implemented in the variable space, would be an almost ideal choice for diversity characterization. However, the required determination of the *nearest-neighbor Euclidian distances* for every member of the population is likely to become computationally prohibitive in the case of high dimensional optimization problems. A novel diversity characterization/metric is developed in this paper. Salient features of this metric are:

- It seeks to effectively capture the two diversity attributes: the overall spread and the distribution of particles.
- It is computationally inexpensive to implement, if required, at every iteration.

Separate diversity metrics and diversity preservation mechanisms are formulated for continuous and discrete design variables. The diversity metrics and the corresponding diversity preservation coefficients are estimated for the entire population at the start of an iteration. The diversity metrics are then updated using a common factor that seeks to account for the particle distribution. In the case of continuous design variables, the initial diversity metric is given by the normalized side length of *the smallest hypercube that encloses all the particles*. This metric is expressed as

$$D_c = \left(\prod_{i=m+1}^{n} \frac{x_i^{t,\max} - x_i^{t,\min}}{x_i^{\max} - x_i^{\min}}\right)^{\frac{1}{n-m}} \quad (6)$$

where $x_i^{t,\max}$ and $x_i^{t,\min}$ are respectively the maximum and the minimum values of the ith design variable in the population at the tth iteration; and $x_i^{\max}$ and $x_i^{\min}$, respectively, represent the specified upper and lower bounds of the ith design variable. The parameters n and m represent the total number of variables and the number of discrete variables, respectively. It is important to note that the concept of enclosing hypercube (of the entire population or an elite subset of the population) has been used in different ways to improve the efficiencies of heuristic optimization algorithms. For example, Wang et al. (2009) used the hypercube that encloses a set of elite solutions (surrounding the feasible region) to shrink the search space, and consequently accelerate the feasible region seeking process during constrained evolutionary optimization.

An undesirable and common scenario in heuristic algorithms is the presence of one or more outlier particles, when the majority of the particles are concentrated in a significantly smaller region. Occurrence of this scenario leads to an appreciable overestimation of the population diversity (D_c). To overcome this deleterious scenario, as well as to account for the distribution of candidate solutions, the diversity metric is further modified. A hypercuboid region is first constructed around the best global candidate solution in the overall variable-space (including continuous and discrete variables). This hypercuboid region is defined such that its volume is a *fraction* of the volume of the smallest hypercube enclosing all the particles. The *user-defined fraction* is represented by the parameter λ, where $0 < \lambda < 1$. The "number of particles" within the fractional hypercuboid region is then determined and used to adjust the continuous diversity metric (enclosing-hypercube side length), in order to better account for the particle distribution. The boundaries of the fractional hypercuboid region is given by

$$\begin{aligned} \bar{x}_i^{t,\max} &= \max\begin{bmatrix} x_i^{t,\min} + \lambda\Delta x_i^t \\ \min\left(P_{g,i} + 0.5\lambda\Delta x_i^t,\ x_i^{t,\max}\right) \end{bmatrix}, \\ \bar{x}_i^{t,\min} &= \min\begin{bmatrix} x_i^{t,\max} - \lambda\Delta x_i^t \\ \max\left(P_{g,i} - 0.5\lambda\Delta x_i^t,\ x_i^{t,\min}\right) \end{bmatrix} \\ &\forall\, i = 1, 2, \ldots, n \end{aligned} \tag{7}$$

where $\Delta x_i^t = x_i^{t,\max} - x_i^{t,\min}$; the parameters $\bar{x}_i^{t,\max}$ and $\bar{x}_i^{t,\min}$ respectively represent the upper and the lower boundaries of the fractional domain for the design variable x_i; and $P_{g,i}$ is the i^{th} variable of the best global solution. The adjusted continuous diversity metric $\bar{D}_c$ is then expressed as

$$\bar{D}_c = \left(\lambda\frac{N+1}{N_\lambda+1}\right)^{\frac{1}{n}} \times D_c \tag{8}$$

where N_λ is the number of particles in the λ-fractional domain.

The diversity coefficient, γ_c, for continuous variables is then defined as a function of the continuous diversity metric, which is given by

$$\begin{aligned} \gamma_c &= \gamma_{c0}\exp\left(\frac{-\bar{D}_c^2}{2\sigma_c^2}\right), \quad \text{where} \\ \sigma_c &= \frac{1}{\sqrt{2\ln\left(1/\gamma_{\min}\right)}} \end{aligned} \tag{9}$$

and γ_{c0} and $\gamma_{\min}$ are specified constants that respectively control the scale of the diversity coefficient and the variance of the diversity coefficient with the diversity metric. The order of magnitude of the diversity-scaling constant γ_{c0} should be one; or, in other words, it should be comparable to that of the explorative coefficient, β_g. In the range 0 to 1 for $\bar{D}_c$, the diversity coefficient is a monotonically decreasing function. The nature of this function for different orders of magnitude of $\gamma_{\min}$ is shown in Fig. 4.

In the case of discrete design variables, the diversity is characterized independently for each variable in order to address the following two factors:

i. The effective diversity in the i^{th} discrete variable depends on (1) the number of feasible values available for that variable and (2) the distribution of these feasible values.

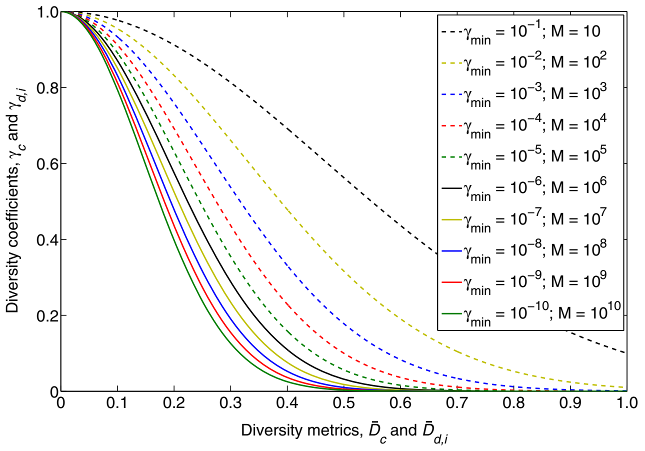


Fig. 4 Variation of the diversity coefficients γ_c and $\gamma_{d,i}$ with the diversity metrics $\bar{D}_c$ and $\bar{D}_{d,i}$, respectively, illustrated at (i) different values of $\gamma_{\min}$ for continuous variables, and (ii) different sizes (M) of the feasible set for discrete variables, with $\gamma_{d0} = 1$

ii. Diversity preservation in discrete variables should seek to avoid the stagnation of particles inside a local discrete-space hypercube H_d.

The initial diversity metric (D_d) for discrete design variables is a vector of the *normalized discrete variable ranges* that span the current population. This metric is expressed as

$$D_{d,i} = \frac{x_i^{t,\max} - x_i^{t,\min}}{x_i^{\max} - x_i^{\min}}, \quad \forall\, i = 1, 2, \ldots, m \tag{10}$$

where $D_{d,i}$ is the component of the discrete diversity metric corresponding to the i^{th} discrete variable. Subsequently, in order to better account for the *distribution of solutions*, the discrete diversity metric is adjusted as

$$\bar{D}_{d,i} = \left(\lambda\frac{N+1}{N_\lambda+1}\right)^{\frac{1}{n}} \times D_{d,i} \tag{11}$$

where $\bar{D}_{d,i}$ is the adjusted discrete diversity metric. It is important to note how the parameter λ couples the diversity in continuous and discrete design variables. As a result, the diversity preservation mechanisms for continuous and discrete variables are expected to work in coherence with each other.

Diversity preservation for discrete variables is accomplished through modification of the discrete update process described in Section 2.2. The otherwise deterministic approximation of the particle to a nearby feasible discrete location is replaced by a stochastic update process. This stochastic update gives a particle the opportunity to jump out of a local hypercube, thereby reducing the possibility of stagnation of the swarm's discrete component. A vector of discrete-variable diversity coefficients, γ_d, is defined to further regulate the updating of discrete variables, with the objective to minimize the possibility of their premature stagnation. A random number (r_4) is generated between 0 and 1, and the stochastic update for the generic i^{th} discrete variable (x_i) of a particle is then applied using the following rules:

i. If r_4 is greater than the diversity coefficient $\gamma_{d,i}$, then update the discrete variable using (4).
ii. If r_4 is less than or equal to $\gamma_{d,i}$, then randomly approximate x_i to either x_i^L or x_i^U (defined in (4)).

The discrete-variable diversity coefficient, $\gamma_{d,i}$, that regulates the stochastic update rules is designed to adapt to the size of *the set of feasible values* for the i^{th} discrete variable. This approach avoids a false impression of considerable diversity, in the case of discrete variables that take a relatively small sized *set of feasible values*. The discrete diversity coefficient is defined as

$$\begin{aligned} &\gamma_{d,i} = \gamma_{d0} \exp\left(\frac{-\bar{D}_{d,i}^2}{2\sigma_{d,i}^2}\right), \quad \text{where} \\ &\sigma_{d,i} = \frac{1}{\sqrt{2\ln M_i}} \\ &\forall\, i = 1, 2, \ldots, m \end{aligned} \tag{12}$$

and where M_i represents the size of the *set of feasible values* for the i^{th} discrete variable, and γ_{d0} is a prescribed constant between 0 and 1. For any estimated value of the population diversity, a higher value of the prescribed parameter, γ_{d0}, makes the random update of the discrete domain location more likely.

It is important to note that, while the continuous-variable diversity coefficient (γ_c) directly regulates the particle motion (in the location update step), the discrete-variable diversity coefficients ($\gamma_{d,i}$) control the updating of the discrete variables as a post-process (during the NVA application) in every pertinent iteration. In addition, the same value of γ_c is used for all design variables at a particular iteration, whereas a different value of $\gamma_{d,i}$ is used for each generic i^{th} discrete variable. An illustration of the discrete diversity coefficient for different sizes of the *set of feasible values* is shown in Fig. 4.

4 Numerical experiments

To validate the Mixed-Discrete Particle Swarm Optimization (MDPSO) algorithm, we apply it to two different classes of single-objective optimization problems: (i) standard unconstrained problems, most of which are multimodal, and (ii) Mixed-Integer Non-Linear Programming (MINLP) problems. The MDPSO algorithm is also applied to a large scale real life engineering problem: wind farm optimization. These three sets of numerical experiments are discussed in the following three sub-sections. The values of the prescribed MDPSO parameters for the three sets of numerical experiments are given in Table 1.

4.1 Unconstrained standard optimization problems

The new MDPSO algorithm is applied to a set of nine *standard unconstrained nonlinear optimization test problems with only continuous variables* to compare its performance with that of the basic PSO. For a majority of these test problems, the basic PSO is expected to offer an effective solution. The MDPSO is specifically designed to address complex *constrained and/or mixed-discrete optimization* problems. With this set of numerical experiments, we particularly investigate whether the new MDPSO features related to diversity preservation, introduce any unexpected characteristics. The first eight test problems have been borrowed from the list of sample single objective optimization problems provided in the MATLAB Genetic and Evolutionary Algorithm Toolbox (GEATbx) Documentation (Pohlheim

Table 1 User-defined constants in PSO

Parameter	Unconstrained problems	MINLP	Wind farm optimization
α	0.5	0.5	0.5
β_g	1.4	1.4	1.4
β_l	1.4	1.4	1.4
γ_{c0}	0.1, 0.5, 1.0	2.0	5.0
γ_{d0}	–	0.7	1.0
$\gamma_{\min}$	1.0e−10	1.0e−10	1.0e−05
Population size (N)	$10 \times n$	$10 \times n$	$20 \times n$
Fractional domain size ($\lambda \times N$)	$0.25 \times N$	$0.1 \times N$	$0.1 \times N$
Allowed number of function calls	10,000	50,000	600,000

Table 2 Standard unconstrained optimization problems

Test problem	Function name	Number of variables	Complexity attribute
1	Rosenbrock's valley	2	Long relatively flat valley
2	Rastrigin's function	2	Highly multimodal
3	Schwefel's function	2	Highly multimodal
4	Griewangk's function	2	Highly multimodal
5	Ackley's path function	2	Highly multimodal
6	Michalewicz's function	10	Flat regions and multimodal
7	Easom's function	2	Mostly flat search space
8	Goldstein–Price's function	2	Extensive flat region
9	Miele–Cantrell	4	Multimodal

2010). The GEATbx problems were originally developed and reported by different researchers from the design and optimization community. The last test problem from Table 2 (Miele–Cantrell function) has been borrowed from the paper by Miele and Cantrell (1969). Details of the standard unconstrained test problems are summarized in Table 2.

The MDPSO algorithm is applied to each test problem, using three different values of the diversity coefficient scaling constant: $\gamma_{c0} = 0.1, 0.5, 1.0$. Each test problem is run 10 times, with a particular γ_{c0} value, to compensate for the effects of the random operators on the overall algorithm performance. Results of the conventional PSO was obtained by specifying the diversity coefficient scaling constant, γ_{c0}, to be zero, while other basic PSO parameters were fixed at the same values as given in Table 1. The algorithms are terminated when the best global solution does not improve by at least 1.0e−10 times its objective value in 10 consecutive iterations. The convergence histories for the Miele Cantrell test function from representative runs of the MDPSO and a representative run of the conventional PSO are shown in Fig. 5. The actual minimum objective value for this test function is 0.0. It can be observed from Fig. 5 that the algorithms perform very well for the multimodal Miele–Cantrell test function. With the diversity scaling constant equal to 0.1 (black dashed line), the rate of convergence of MDPSO is approximately twice that of the conventional PSO—the objective function reduces to 1.0e−07 in half the number of function calls. With the diversity scaling constant equal to 1.0 (grey long-dashed line), the MDPSO algorithm converges slightly slower than the conventional PSO algorithm. This phenomenon can be attributed to the increased reduction in the particle velocities towards the global optimum, caused by the preservation of a larger amount of population diversity among the particles.

For each test problem in Table 2, the actual minimum of the objective function is known. Using the actual minimum objective function value, a normalized relative error is evaluated to represent the optimization performance. This normalized relative error (ε_f) is expressed as

$$\varepsilon_f = \begin{cases} \dfrac{\left|f_{\min}^{\text{comp}} - f_{\min}^{\text{act}}\right|}{f_{\min}^{\text{act}}}, & \text{if } f_{\min}^{\text{act}} \neq 0 \\ \left|f_{\min}^{\text{comp}} - f_{\min}^{\text{act}}\right|, & \text{if } f_{\min}^{\text{act}} = 0 \end{cases} \tag{13}$$

where $f_{\min}^{\text{comp}}$ and $f_{\min}^{\text{act}}$ are the computed minimum and the actual minimum of the objective function, respectively. The normalized relative errors given by the best and the worst optimized solutions among the 10 runs of each test problem are shown in Fig. 6a and b. Figure 6a specifically shows the resulting errors when conventional PSO and MDPSO are applied to the same test problems. Figure 6b specifically shows the resulting errors when MDPSO is applied to the test problems using different values of the specified diversity scaling constant (γ_{c0}). Further details, regarding the

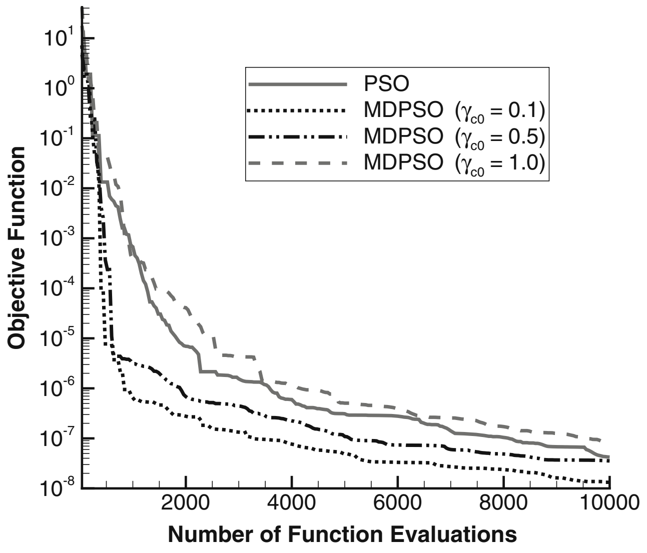
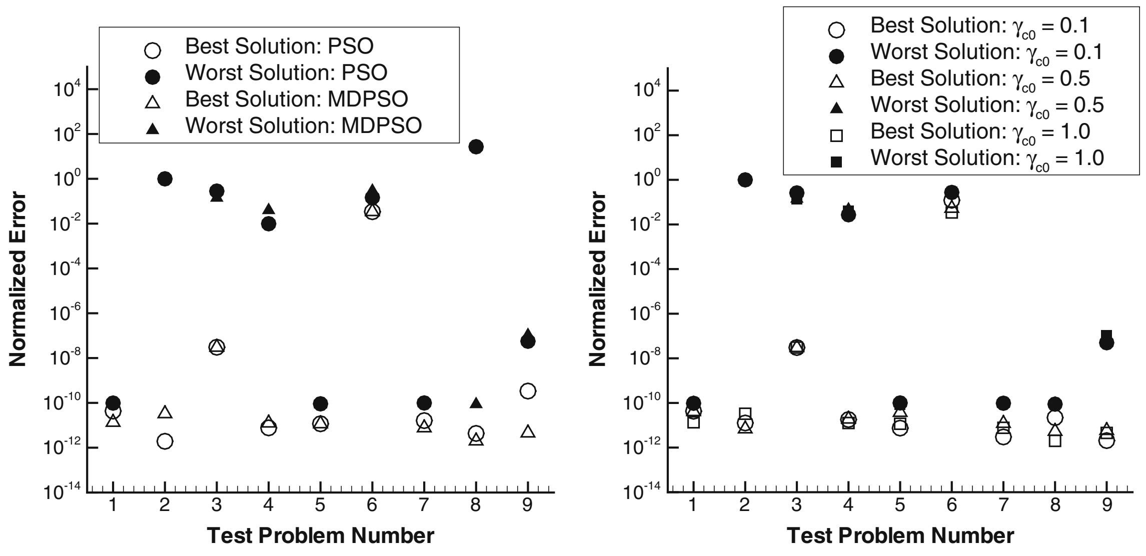


Fig. 5 Convergence histories of the MDPSO and the conventional PSO for the Miele–Cantrell function



(a) Using PSO, and MDPSO with γ_{c0} = 1.0

(b) Using MDPSO with γ_{c0} = 0.1, 0.5, 1.0

Fig. 6 Maximum and minimum values (among 10 runs) of the normalized relative error obtained for the standard unconstrained test problems

performance of the MDPSO algorithm with the diversity scaling constant equal to 1.0, are provided in Table 3.

Figure 6a shows that the MDPSO algorithm performs as well as or better than the conventional PSO algorithm for most of the standard unconstrained test problems, except for test problem 2. It is observed from Fig. 6a and Table 3 that neither of the algorithms could provide satisfactory solutions for test problem 6, Michalewicz's function; these observations can be attributed to the existence of extensive flat regions and multimodality in the Michalewicz's function (Pohlheim 2010). Overall, Fig. 6a illustrates that the additional diversity preservation features in MDPSO does not introduce any undesirable characteristics into the dynamics of the PSO algorithm. Figure 6b illustrates that the performance of MDPSO is marginally sensitive to the specified value of the diversity scaling constant (γ_{c0}) in the case of these unconstrained continuous problems—the relative errors given by MDPSO for the three different values of the diversity scaling constant are close to each other. The standard deviation in the computed minima obtained from the 10 runs for each test problem (Table 3) is observed to be relatively small when compared to the corresponding actual minima. This observation further illustrates the consistency in the performance of the MDPSO algorithm.

4.2 Mixed-Integer Nonlinear Programming (MINLP) problems

The MDPSO algorithm is applied to an extensive set of ninety-eight Mixed-Integer Non-Linear Programming (MINLP) test problems; these test problems were obtained from the comprehensive list of one hundred MINLP problems reported by Schittkowski (2009). The problems

Table 3 Performance of MDPSO (with $\gamma_{c0} = 1.0$) on the standard unconstrained test problems

Test problem	Actual minimum	Best computed minimum	Worst computed minimum	Standard deviation of computed minima
1	0.000E+00	1.359E−11	9.951E−11	3.274E−11
2	0.000E+00	3.283E−11	9.950E−01	5.138E−01
3	−8.380E+02	−8.380E+02	−7.195E+02	6.242E+01
4	0.000E+00	1.263E−11	3.946E−02	1.269E−02
5	0.000E+00	1.167E−11	8.373E−11	2.276E−11
6	−9.660E+00	−9.328E+00	−6.843E+00	8.119E−01
7	−1.000E+00	−1.000E+00	−1.000E+00	2.723E−11
8	3.000E+00	3.000E+00	3.000E+00	8.614E−11
9	0.000E+00	4.534E−12	1.054E−07	4.436E−08

numbered 10 and 100 in the original list (Schittkowski 2009) have not been tested in this paper. A majority of these MINLP test problems belong to the GAMS Model Library MINLPlib (Bussieck et al. 2007), and have been widely used to validate and compare optimization algorithms (Schittkowski 2009). These MINLP test problems present a wide range of complexities:

- the total number of design variables varies from 2 to 50;
- the numbers of binary design variables and integer design variables vary from 0 to 16 and 0 to 50, respectively;
- the total number of constraints (including equality and inequality) varies from 0 to 54;
- the number of equality constraints varies from 0 to 17.

Similar to the previous set of numerical experiments, each MINLP test problem is run 10 times to compensate for the performance effects of the random operators in the algorithm. For each run, the algorithm is terminated when the best global solution does not improve by at least 1.0e−06 times its objective value in 10 consecutive iterations.

For ease of illustration and ready interpretation of the results obtained by MDPSO, we divide the set of 98 MINLP test problems into six classes based on the number of design variables and the presence of continuous variables. These MINLP problem classes are shown in Table 4. The normalized relative errors corresponding to the best and the worst solutions among the 10 runs obtained (by MDPSO) for each test problem from the six classes are illustrated separately in Fig. 7a–f. These figures also show the number of design variables (solid gray line) and the number of constraints (dashed gray line) in each test problem, to help understand their impact on the optimization performance. A histogram of the relative errors is shown in Fig. 8. It is helpful to note that, in the case of several *purely integer test problems*, a zero relative error is obtained through optimization. In order to allow a logarithmic scale illustration of the errors, these zero errors are replaced by an artificial error value of 1.0e−12 in the figures.

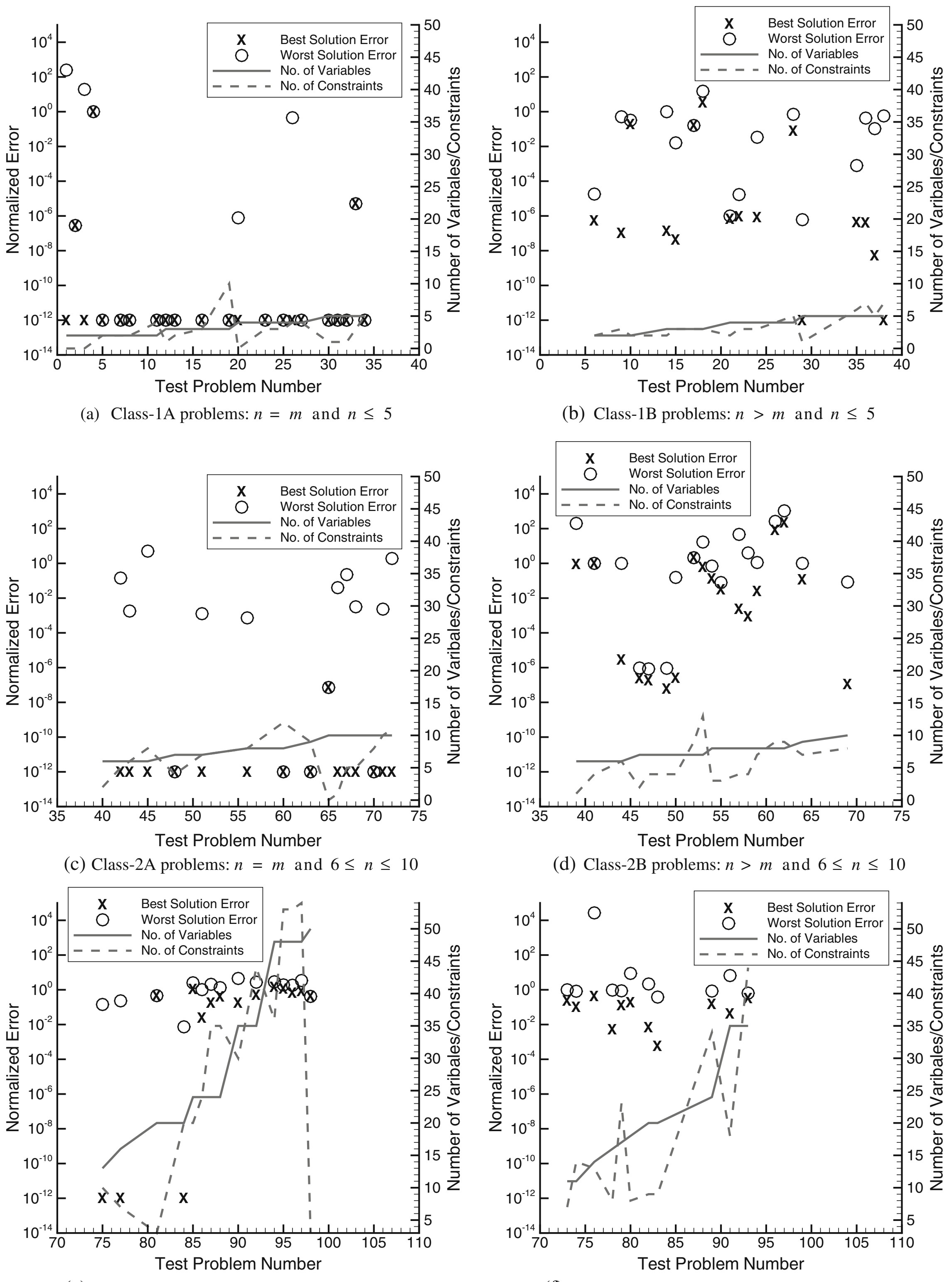
Figure 7a and b show that MDPSO performs significantly better for the *purely integer* problems (Class-1A) than for the mixed-integer problems (Class-1B). This observation can be partly attributed to the practical possibility of finding the exact minimum (no error) if the variables are all integers instead of continuous real numbers. Overall, a majority of the Class-1 test problems (with $n \leq 5$) are observed to have converged to relative errors less than 1.0e−04. Similarly, for Class-2 problems, Fig. 7c and d show that MDPSO performs better in the case of *purely integer* problems (Class-2A). Interestingly, for the Class-2A problems, the best solution among 10 runs is observed to mostly converge to the exact minima ($\varepsilon_f = 1.0\text{e}{-12}$ in the figures). It is observed that, for a majority of the Class-3 problems (Fig. 7e and f), MDPSO has not converged beyond a relative error of 1.0e−04, which can be primarily attributed to the high number of constraints in these MINLP problems. Some of the Class-3 problems have more than 30 constraints. A higher number of design variables ($11 \leq n \leq 50$) might also have partly contributed to the lack of convergence in the Class-3 problems.

The histogram in Fig. 8 illustrates the distribution of the relative error, on a logarithmic scale, for all the 10×98 test problem runs. It is observed that the MDPSO algorithm has converged to acceptable relatively errors of less than 1.0e−03 in approximately 50 % of the test problem runs. It is important to note that the same prescribed parameter values were specified for all MINLP test problems (Table 1) for a fair illustration of algorithm performance. The set of MINLP test problems present a wide variety of nonlinear criteria functions and problem complexities, which ideally demands different parameter values to be specified for characteristically different MINLP problems. The development of general guidelines regarding how to specify the prescribed MDPSO parameter values for different types of MINLP problems is therefore an important topic for further research.

Figure 9a and b illustrate the net constraint violation (5) corresponding to the best and the worst solutions obtained by MDPSO, respectively for MINLP problems without and with equality constraints. It is observed from Fig. 9a that the MDPSO algorithm has successfully found the feasible space in a majority of the MINLP problems without inequality constraints; the handful of exceptions is observed to generally involve more than 35 design variables and 35 constraints. The feasibility success of MDPSO is lower for the MINLP problems with equality constraints than those without equality constraints (Fig. 9b). Overall, MDPSO found the feasible region with the best solution in a majority of the test problems (90 out 98). Cabrera and Coello (2007) stated that significant advancement is necessary to extend the application of standard PSO to solve *constrained* optimization problems. From that perspective, the performance of MDPSO in finding the feasible region seems promising, particularly considering that the concerned test problems are of mixed-discrete nature. However, addressing equality

Table 4 MINLP problem classes

Number of variables (n)	Without continuous variables ($n = m$)	With continuous variables ($n > m$)
$2 \leq n \leq 5$	Class-1A	Class-1B
$6 \leq n \leq 10$	Class-2A	Class-2B
$11 \leq n \leq 50$	Class-3A	Class-3B



(a) Class-1A problems: $n = m$ and $n \leq 5$

(b) Class-1B problems: $n > m$ and $n \leq 5$

(c) Class-2A problems: $n = m$ and $6 \leq n \leq 10$

(d) Class-2B problems: $n > m$ and $6 \leq n \leq 10$

(e) Class-3A problems: $n = m$ and $11 \leq n \leq 50$

(f) Class-3B problems: $n > m$ and $11 \leq n \leq 50$

Fig. 7 Normalized relative errors for the best and the worst solutions (among 10 runs) obtained by MDPSO for the MINLP problems

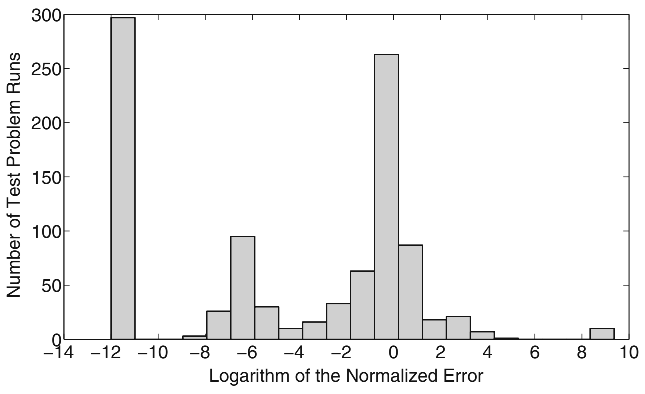


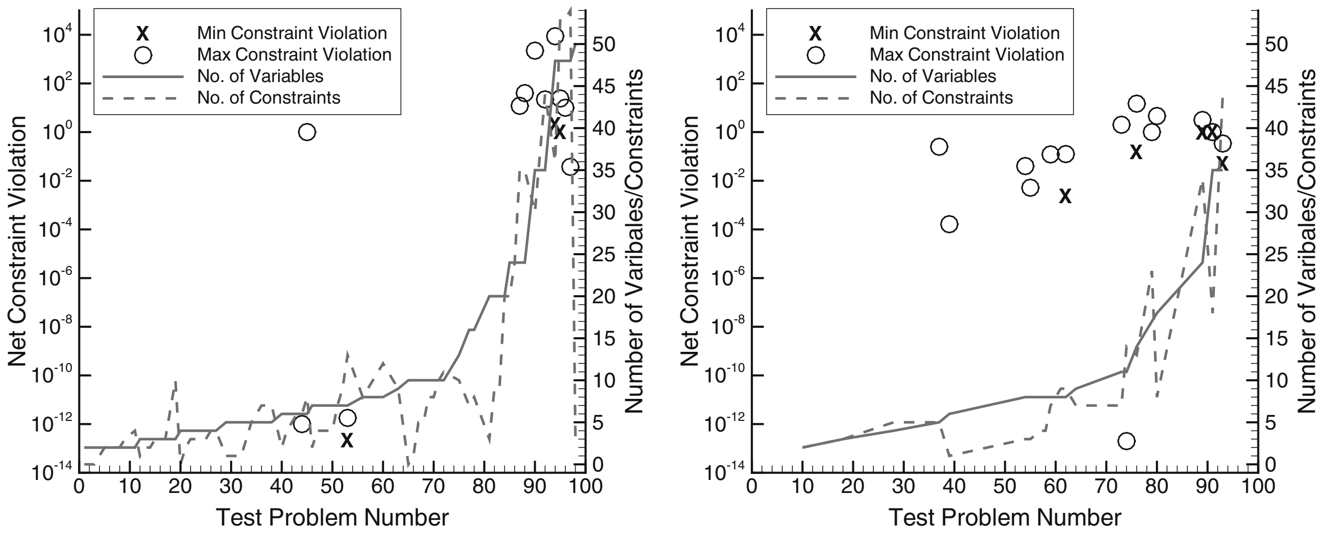
Fig. 8 Histogram of the *order of magnitude* of the normalized relative error obtained by MDPSO for the MINLP test problems

constraints in PSO, in the presence of mixed-discrete variables, remains a major challenge.

In addition to the ability to successfully find the feasible space and the optimum design, the number of function evaluations invested in the process is also an important performance attribute for an optimization algorithm. Figure 10a–c illustrate the number of function evaluations made by the MDPSO algorithm for Class-1, Class-2, and Class-3 MINLP problems, respectively. A majority of the Class-1 and Class-2 MINLP problems converged in less than 30,000 function evaluations. A significant number of the Class-1 problems required less than 1000 function evaluations to converge. In contrast, most of the Class-3 problems exhausted the maximum allowed number of function evaluations (50,000). This greater computational expense can be attributed to the significant investment necessary in finding the feasible region for these constrained high dimensional MINLP problems.

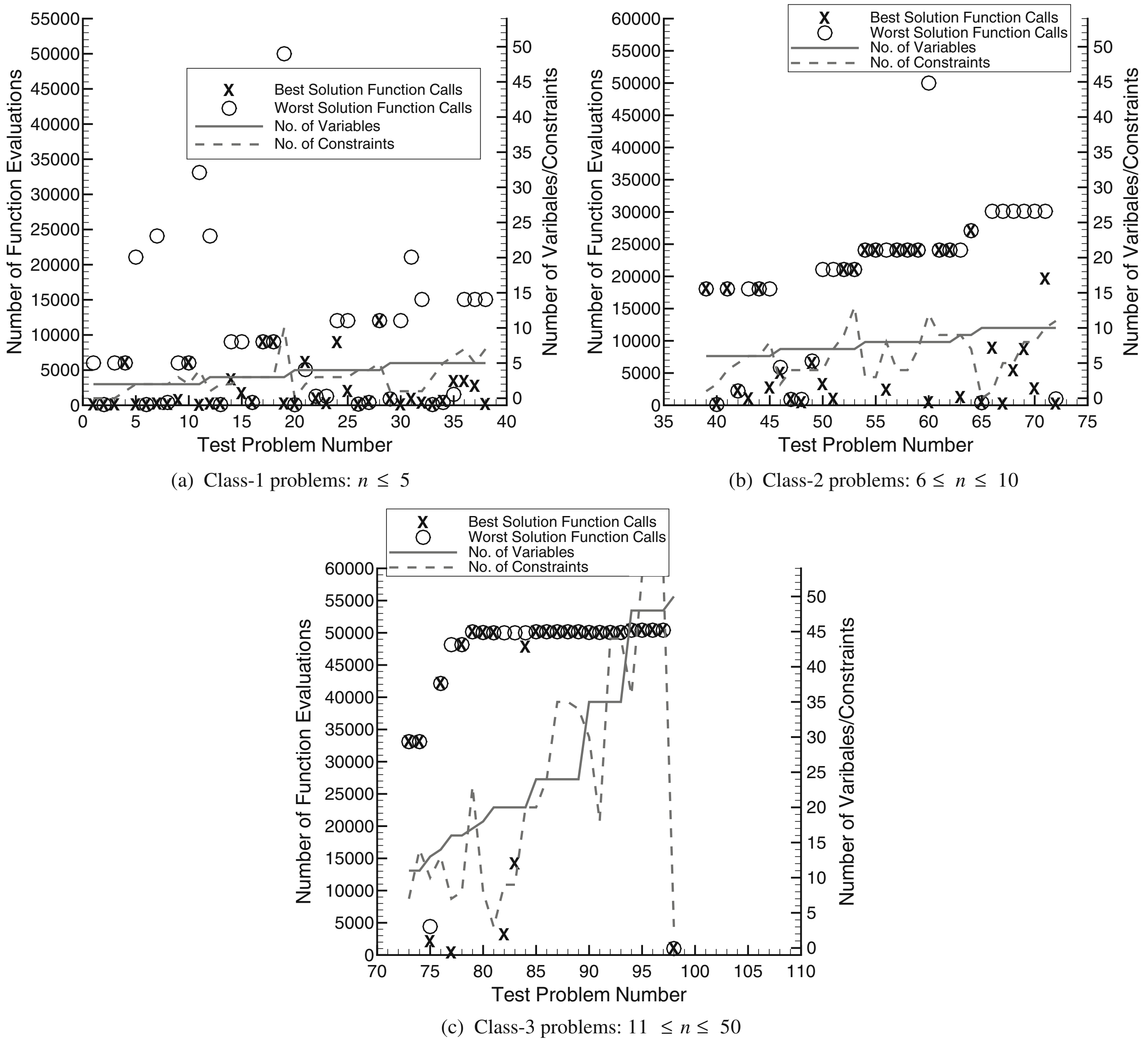
Further details regarding the performance of the MDPSO algorithm for the MINLP problems are provided in Tables 5, 6, and 7. The results of the MDPSO algorithm (for the MINLP problems) are also compared with those of a popular binary genetic algorithm—the Non-dominated Sorting Genetic Algorithm (bin-NSGA-II) (Deb et al. 2002). Although NSGA-II is typically suited for multi-objective problems, it has been used to solve constrained single objective problems (Murugan et al. 2009). NSGA-II is a popular choice among heuristic algorithms, when solving MINLP problems. In the current set of numerical experiments, NSGA-II is also run ten times for each MINLP problem. The population is set as 10 times the number of design variables and the maximum allowed number of function evaluations is 50,000; both of these specifications are the same as prescribed for MDPSO (Table 1). In the case of NSGA-II, the crossover probability is set at 0.8, and the string length for each variable is set at 8. NSGA-II is applied to solve 46 problems out of the entire suite of 98 MINLP problems; at least two problems are solved from each Class of MINLP problems. The results of NSGA-II are included in Tables 5–7.

In Tables 5–7, the *feasibility success %* represents the fraction of the 10 runs for which the concerned optimization algorithm found the feasible region. The rows that appear light gray represent MINLP problems for which the *best minimum function value (among 10 runs)* found by MDPSO was worse than that found by NSGA-II. The rows that



(a) MINLP problems *without* equality constraints (b) MINLP problems *with* equality constraints

Fig. 9 Net constraint violation in the best and the worst solutions (among 10 runs) obtained by MDPSO for MINLP problems



(a) Class-1 problems: $n \leq 5$

(b) Class-2 problems: $6 \leq n \leq 10$

(c) Class-3 problems: $11 \leq n \leq 50$

Fig. 10 Number of function evaluations used for the best and the worst solutions (among 10 runs) obtained by MDPSO for the MINLP problems

appear dark gray represent MINLP problems for which the *standard deviation of the minimum function values (among 10 runs)* found by MDPSO was worse than that found by NSGA-II. In the column headings for these three tables, "std. dev." means standard deviation.

It is observed from Tables 5–7 that MDPSO finds the feasible region with 100 % success in a majority of the MINLP problems. The feasibility success of NSGA-II is in general observed to be lower than that of MDPSO, which can be partly attributed to an expected slower rate of convergence of NSGA-II for single objective problems. MDPSO also outperforms NSGA-II for the majority of the Class-1 MINLP problems, both in terms of the best computed minimum and the standard deviation of the computed minima (out of 10 runs). However, in the cases of Class-2 and Class-3 MINLP problems, the performances of MDPSO and NSGA-II seem comparable. Future work should therefore seek to further advance the MDPSO algorithm for more robust application to high-dimensional and highly constrained mixed-discrete optimization problems.

4.3 Wind farm optimization problem

In this section, we present the application of the MDPSO algorithm to a complex real life engineering design

Table 5 Performance of MDPSO and NSGA-II on the Class-1 MINLP test problems

Class	No.	Feasibility success % (MDPSO)	Feasibility success % (NSGA-II)	Actual minimum	Best computed minimum (MDPSO)	Best computed minimum (NSGA-II)	Std. dev. of computed minima (MDPSO)	Std. dev. of computed minima (NSGA-II)
1A	1	100	100	7.200E−01	7.200E−01	2.452E+01	6.860E+01	5.090E+03
	2	100	100	1.770E+00	1.770E+00	1.770E+00	2.340E−16	2.341E−16
	3	100	100	7.031E−01	7.030E−01	7.031E−01	6.520E+00	6.911E+00
	4	100	100	−6.000E+00	0.000E+00	0.000E+00	0.000E+00	3.112E+09
	5	100	100	−5.694E+01	−5.690E+01	−5.694E+01	0.000E+00	5.929E−02
	7	100	100	1.600E+01	1.600E+01	1.600E+01	0.000E+00	4.216E−01
	8	100	100	−3.108E+02	−3.110E+02	−3.108E+02	5.990E−14	1.265E+00
	11	100	100	3.100E+01	3.100E+01	3.100E+01	0.000E+00	0.000E+00
	12	100	100	1.000E+00	1.000E+00	1.000E+00	0.000E+00	4.830E−01
	13	100	100	4.000E+00	4.000E+00	4.000E+00	0.000E+00	4.029E+01
	16	100	100	−4.310E+02	−4.310E+02	−4.310E+02	0.000E+00	3.678E+01
	19	100	100	−8.050E+01	−8.050E+01	−8.050E+01	0.000E+00	3.300E+00
	20	100	100	1.000E+00	1.000E+00	1.000E+00	3.160E−07	0.000E+00
	23	100	100	−3.800E+01	−3.800E+01	−3.800E+01	0.000E+00	0.000E+00
	25	100	–	2.000E+00	2.000E+00	–	0.000E+00	–
	26	100	100	−2.000E+01	−2.000E+01	−2.000E+01	3.750E+00	0.000E+00
	27	100	100	−4.812E+02	−4.810E+02	−4.780E+02	5.990E−14	1.708E+00
	30	100	100	2.810E+02	2.810E+02	2.810E+02	0.000E+00	0.000E+00
	31	100	100	−4.500E+03	−4.500E+03	−4.500E+03	0.000E+00	0.000E+00
	32	100	100	8.000E+00	8.000E+00	8.000E+00	0.000E+00	7.554E+00
	33	100	100	1.004E−01	1.000E−01	1.004E−01	1.460E−17	4.242E−03
	34	100	80	−5.852E+02	−5.850E+02	−5.852E+02	1.200E−13	8.509E+00
1B	6	100	100	−2.444E+00	−2.440E+00	−2.444E+00	1.590E−05	5.218E−04
	9	100	–	−2.460E+02	−1.990E+02	–	1.020E+01	–
	10	100	100	−1.700E+01	−1.700E+01	−1.700E+01	4.190E+00	3.165E+00
	14	100	100	−5.685E+00	−5.680E+00	−5.635E+00	2.780E+00	1.266E+00
	15	100	–	1.247E+01	5.280E+01	–	5.980E+01	–
	17	100	100	2.345E+01	2.340E+01	4.605E+01	1.200E−01	7.606E+02
	18	100	100	1.077E+00	1.250E+00	1.081E+00	0.000E+00	6.165E−02
	21	100	40	1.363E+00	1.360E+00	1.616E+00	2.340E−16	7.683E−01
	22	100	100	−4.096E+01	−4.100E+01	−4.087E+01	2.270E−04	2.339E+00
	24	100	100	7.198E+03	7.200E+03	7.228E+03	9.670E+01	3.940E+02
	28	100	–	2.824E+01	3.040E+01	–	5.960E+00	–
	29	100	100	−1.001E+04	−1.000E+04	−1.001E+04	4.220E−03	2.647E+00
	35	90	–	7.667E+00	7.670E+00	–	3.810E−01	–
	36	100	100	−3.067E+04	−3.070E+04	−3.040E+04	9.510E+00	1.739E+02
	37	100	100	3.500E+00	3.500E+00	3.504E+00	7.750E−01	1.095E+00
	38	100	100	3.500E+00	3.500E+00	3.500E+00	8.560E−01	8.486E−01

problem: wind farm optimization. This problem presents (i) a significantly large number of design variables: 50 continuous variables, and 25 discrete variables, and (ii) a highly nonlinear and multimodal objective function. At the same time, the objective function evaluation for this problem is computationally expensive. A computationally efficient yet robust optimization methodology needs to be leveraged to address such a challenging problem.

Wind farms consist of multiple wind turbines located in a particular arrangement over a substantial stretch of

Table 6 Performance of MDPSO and NSGA-II on the Class-2 MINLP test problems

Class	No.	Feasibility success % (MDPSO)	Feasibility success % (NSGA-II)	Actual minimum	Best computed minimum (MDPSO)	Best computed minimum (NSGA-II)	Std. dev. of computed minima (MDPSO)	Std. dev. of computed minima (NSGA-II)
2A	40	100	100	−9.250E+00	−2.150E+10	−2.147E+10	0.000E+00	0.000E+00
	42	100	–	−7.000E+00	−7.000E+00	–	3.160E−01	–
	43	100	80	−7.784E+02	−7.780E+02	−7.784E+02	6.760E−01	6.152E+00
	45	90	70	1.122E+05	1.120E+05	1.522E+05	1.780E+05	6.451E+04
	48	100	–	7.000E+02	7.000E+02	–	0.000E+00	–
	51	100	–	−1.100E+03	−1.100E+03	–	6.760E−01	–
	56	100	–	−1.098E+03	−1.100E+03	–	3.370E−01	–
	60	100	–	2.300E+00	2.300E+00	–	4.680E−16	–
	63	100	–	−1.125E+03	−1.130E+03	–	2.400E−13	–
	65	100	–	−4.313E+01	−4.310E+01	–	0.000E+00	–
	66	100	100	3.000E+02	3.000E+02	3.000E+02	3.780E+00	6.637E+01
	67	100	–	4.710E+02	4.710E+02	–	5.170E+01	–
	68	100	100	−1.281E+01	−1.280E+01	−1.271E+01	2.070E−02	7.034E−01
	70	100	–	4.300E+01	4.300E+01	–	0.000E+00	–
	71	100	–	−1.033E+03	−1.030E+03	–	7.590E−01	–
	72	100	–	−1.100E+02	−1.100E+02	–	7.700E+01	–
2B	39	70	–	1.000E+00	1.890E+00	–	7.830E+01	–
	41	100	90	−4.574E+03	−8.000E+00	−4.000E+00	0.000E+00	0.000E+00
	44	100	70	6.010E+00	6.010E+00	6.031E+00	2.290E+00	8.743E−01
	46	100	–	1.000E+00	1.000E+00	–	3.160E−07	–
	47	100	–	6.949E+02	6.950E+02	–	1.580E−04	–
	49	100	–	−2.718E+00	−2.720E+00	–	7.380E−07	–
	50	100	–	−8.980E+06	−8.980E+06	–	4.600E+05	–
	52	100	–	4.580E+00	1.430E+01	–	0.000E+00	–
	53	90	–	−3.339E+02	−1.260E+02	–	1.580E+03	–
	54	60	–	5.964E+00	6.740E+00	–	1.070E+00	–
	55	20	–	−4.036E+04	−3.910E+04	–	6.200E+02	–
	57	100	–	1.000E+00	1.000E+00	–	1.510E+01	–
	58	100	–	1.000E+00	1.000E+00	–	1.290E+00	–
	59	70	–	2.925E+00	3.000E+00	–	1.330E+00	–
	61	30	–	5.471E+00	4.470E+02	–	2.940E+02	–
	62	0	–	6.058E+00	1.340E+03	–	2.020E+03	–
	64	100	–	−1.923E+00	−1.700E+00	–	5.380E−01	–
	69	100	–	3.722E+01	3.720E+01	–	9.900E−01	–

land (onshore) or water body (offshore). The net energy produced by a wind farm is significantly reduced by the wake effects—i.e., the shading effect of a wind turbine on other turbines downstream from it (Beyer et al. 1996). This energy loss can be partially regained by optimizing the arrangement of wind turbines in a farm (farm layout). The overall energy production capacity and the cost of the wind farm also depend on the type(s) of wind turbines installed. Traditionally, a wind farm is comprised of a uniform type of wind turbine. Chowdhury et al. (2012) showed that using an optimal combination of *wind turbines with differing rotor diameters* can significantly improve the performance of a wind farm. This research (Chowdhury et al. 2012) laid the foundation for the exploration of the benefits of using multiple types of wind turbines in a farm.

Table 7 Performance of MDPSO and NSGA-II on the Class-3 MINLP test problems

Class	No.	Feasibility success % (MDPSO)	Feasibility success % (NSGA-II)	Actual minimum	Best computed minimum (MDPSO)	Best computed minimum (NSGA-II)	Std. dev. of computed minima (MDPSO)	Std. dev. of computed minima (NSGA-II)
3A	75	100	50	−7.000E+00	−7.000E+00	4.295E+09	4.830E−01	0.000E+00
	77	100	100	1.300E+01	1.300E+01	1.300E+01	9.490E−01	0.000E+00
	81	100	–	6.853E−01	1.000E+00	–	0.000E+00	–
	84	100	70	−2.059E+01	−2.060E+01	−1.956E+01	4.370E−02	7.983E−01
	85	100	–	−2.961E+04	3.370E+03	–	1.370E+04	–
	86	100	–	8.300E+00	8.500E+00	–	2.630E+00	–
	87	30	–	1.960E+01	2.300E+01	–	1.290E+01	–
	88	80	–	8.600E+00	1.200E+01	–	3.130E+00	–
	90	50	–	1.030E+01	1.210E+01	–	1.750E+01	–
	92	90	–	1.030E+01	1.550E+01	–	7.170E+00	–
	94	0	–	1.460E+01	3.590E+01	–	6.870E+00	–
	95	0	–	1.630E+01	3.450E+01	–	3.730E+00	–
	96	10	–	1.630E+01	2.710E+01	–	5.960E+00	–
	97	80	–	4.807E+01	8.770E+01	–	3.390E+01	–
	98	100	–	1.713E+00	1.000E+00	–	0.000E+00	–
3B	73	30	–	−9.435E−01	−7.290E−01	–	3.240E−01	–
	74	100	–	7.304E+01	8.030E+01	–	1.850E+01	–
	76	0	–	2.545E−01	1.460E−01	–	2.170E+03	–
	78	100	100	2.309E+02	2.320E+02	2.450E+02	6.810E+01	1.555E+01
	79	80	–	6.801E+01	7.640E+01	–	1.910E+01	–
	80	20	–	8.462E−01	6.870E−01	–	2.360E+00	–
	82	100	100	1.219E+01	1.210E+01	1.407E+01	1.310E+01	1.650E+01
	83	100	100	5.315E+00	5.310E+00	7.905E+00	7.070E−01	8.809E+00
	89	0	–	1.446E+05	1.230E+05	–	6.950E+04	–
	91	0	–	−1.430E+00	−1.490E+00	–	2.790E+00	–
	93	0	–	1.419E+01	1.880E+01	–	1.580E+00	–

The current use of MDPSO in wind farm optimization is a progressive extension of the use of PSO as an important component of the Unrestricted Wind Farm Layout Optimization (UWFLO) framework developed by Chowdhury et al. (2012). Several wind farm layout optimization methodologies exist in the literature (Sorensen and Nielsen 2006; Mikkelsen et al. 2007; Beyer et al. 1996; Grady et al. 2005; Sisbot et al. 2010; Gonzalez et al. 2010). The UWFLO methodology (Chowdhury et al. 2012) avoids the limiting assumptions presented by other methods regarding the layout pattern, the selection of turbines, and the variation of wind conditions over time. Optimal selection of wind turbines presents a discrete problem, since only a set of discrete turbine choices is commercially available. This discrete selection scenario was however not considered in the original UWFLO framework (Chowdhury et al. 2012), where turbine rotor-diameters were treated as continuous variables. The current implementation of the MDPSO algorithm in the UWFLO framework allows the type of each turbine to be treated as an integer variable.

Table 8 Specified wind farm properties

Farm property	Value
Location	Baker, ND
Land size (length × breadth)	$(5 \times 7D_0) \times (5 \times 3D_0)$
Orientation	North to South lengthwise
Average roughness	0.1 m (grassland)
Density of air	1.2 kg/m^3

In this paper, the UWFLO method is used to design a rectangular wind farm of specified dimensions, and comprises 25 turbines. A hypothetical wind farm that is located at the Baker wind station in North Dakota (48.167° N, −99.648° W) is considered for this case study. The essential wind-conditions data is obtained from North Dakota Agricultural Weather Network (NDAWN 2010). In the UWFLO method, the wind resource variations are represented using probability distribution models. The specified wind farm properties are given in Table 8; the parameter D_0 in this table represents the average rotor diameter of commercially available wind turbines. The farm is oriented such that the positive X-direction of the layout co-ordinate system points towards the South. Using web data from major turbine manufacturers catering to the US onshore market, an integer number code T^j is assigned to each unique turbine-type-j. A turbine-type is defined by a unique combination of rated-power, rotor-diameter, hub-height, and power characteristics. A list of 66 turbine-types, with rated-powers from 0.6 to 3.6 MW, is prepared and coded from the following turbine manufacturers: GE, Vestas, Gamesa, Siemens, Mitsubishi, and Suzlon.

In this paper, the objective of wind farm optimization is to maximize the net power generation. The net power generation is given by the ratio of "the total energy production over a time period" and "the time period". The location coordinates and the turbine-type of each wind turbine are treated as design variables: 50 continuous variables, and 25 integer variables, respectively. The boundaries of the wind farm, and the minimum distance required between adjacent turbines are treated as inequality constraints. The Cost of Energy (COE) of the wind farm is constrained to a reference cost. This reference cost is the COE, estimated for a reference farm with a 5×5 array layout, a $7D \times 3D$ turbine spacing and comprised of the "GE 1.5 MW xle" turbines (GE-Energy 2009); in this case, D represents the rotor diameter of the reference turbine. Further theoretical and numerical details of the UWFLO model can be found in the paper by Chowdhury et al. (2012).

The farm optimization framework was run several times, and the improvement in farm performance (accomplished through optimization) was found to be consistent across these optimization runs. In this Section, the results from one of the representative optimization runs are illustrated and discussed. An appreciable increase of 60 % in the net power generated by the farm was accomplished through this optimization run. The convergence history for wind farm optimization is shown in Fig. 11. The objective on the Y-axis in Fig. 11 represents the generated farm power normalized by the installed capacity of the reference wind farm. The installed capacity of the farm is 25×1.5 MW. Comparison of the performances of the optimized farm and the reference farm is provided in Table 9.

From Table 9, it is observed that the net power generated by the optimized farm is significantly higher than that generated by the reference farm—a 56.3 % increase. This remarkable increase in power is accomplished at a Cost of Energy lower than that of the reference farm. The optimized

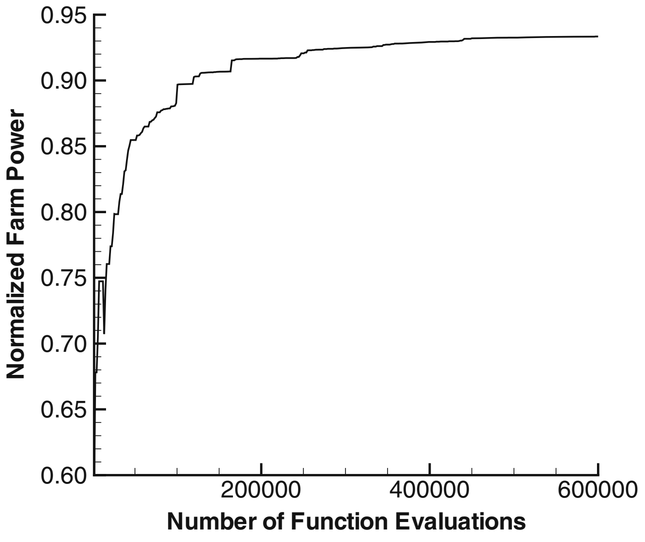


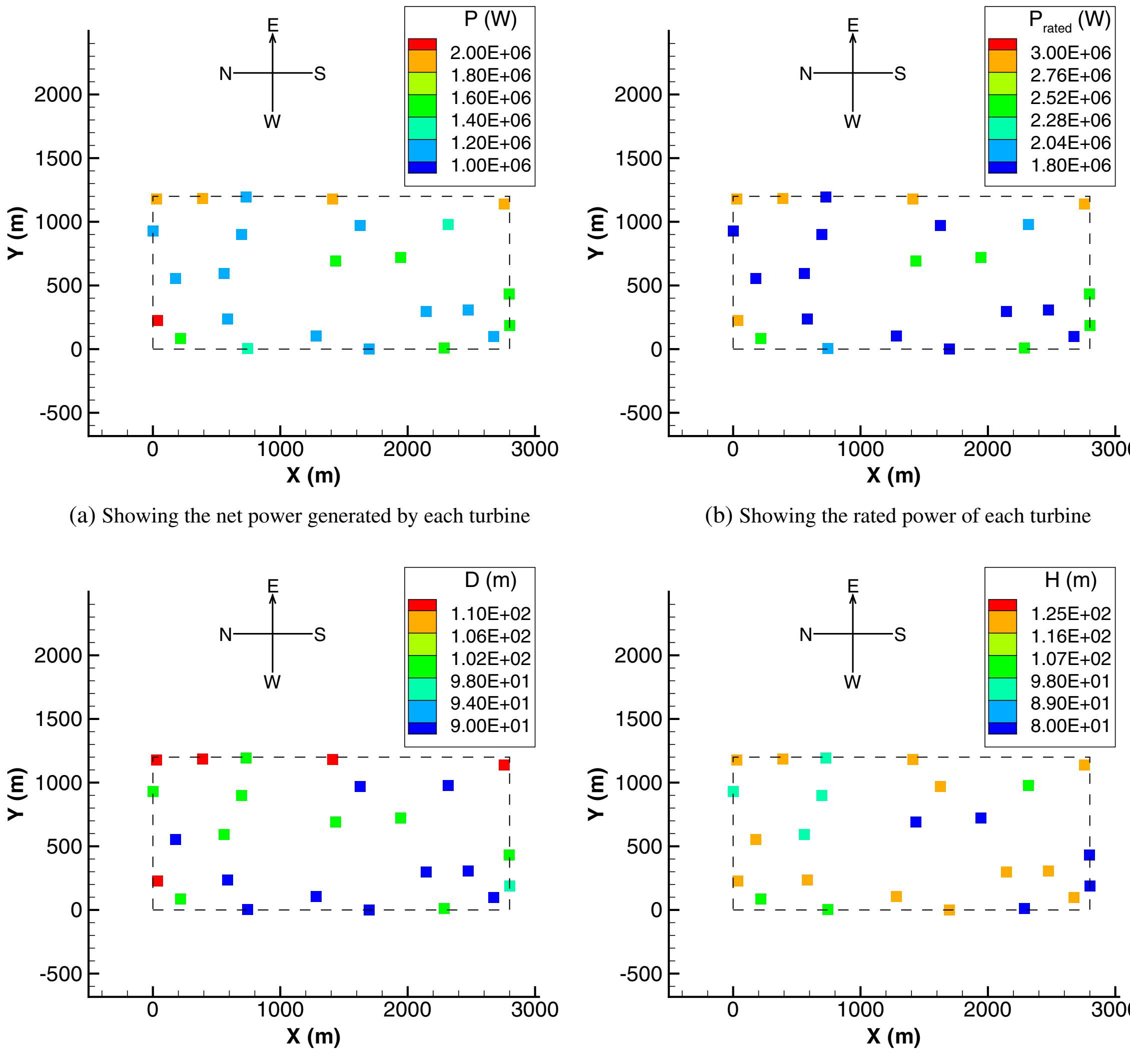
Fig. 11 Convergence history for wind farm optimization

farm is comprised of a combination of eight Vestas 1.8 MW–90 m, four Vestas 1.8 MW–100 m, two Gamesa 2.0 MW–90 m, one Mitsubishi 2.4 MW–95 m, four Mitsubishi 2.4 MW–102 m, one GE 2.5 MW–100 m, and five Vestas 3.0 MW–112 m wind turbines. The overall efficiency of the farm, represented by the power-generated/power-installed ratio, is also observed to be appreciably higher in the case of the optimized farm. Figure 12a–d illustrate the optimized farm layout, where the turbine locations are colored according to (a) the overall power generation, (b) the rated power, (c) the rotor-diameter, and (d) the hub-height of the individual turbines, respectively. The dashed line in these figures represent the farm boundary.

Zhang et al. (2013) have shown that, for this particular site (Baker, ND), the frequency of the wind coming from the East is significantly lower than that coming from the other three cardinal directions. It is indeed interesting to note from Fig. 12b and c that: the UWFLO framework has taken advantage of the turbine selection allowance to place *higher rated-power turbines that come with larger rotor-diameters* on the Eastern edge of the wind farm. These turbines with larger rotor diameters are more suited for lower wind speeds, as experienced by the Eastern face of

Table 9 Performance comparison of the optimized wind farm and the reference wind farm

Parameter	Reference farm	Optimized farm
Normalized power generated	0.597	0.933
Overall farm efficiency	0.597	0.635
COE	0.024	0.023



(a) Showing the net power generated by each turbine

(b) Showing the rated power of each turbine

(c) Showing the rotor-diameter of each turbine

(d) Showing the hub-height of each turbine

Fig. 12 Layout of the optimized wind farm (with optimal combination of turbine-types)

the farm. Their location on the Eastern edge of the farm also ensures minimal effects of the relatively larger wakes created by these turbines.

5 Concluding remarks

A majority of the swarm-based algorithms do not readily address constrained mixed-discrete optimization problems. In this paper, a modification of the Particle Swarm Optimization (PSO) algorithm is developed, which can adequately address system constraints and mixed-discrete variables for single objective problems. Constraint handling is performed using the constrained non-dominance principle adopted from evolutionary algorithms. The conventional particle motion step at each iteration is followed by approximating the discrete component of the variable vector to a neighboring feasible discrete space location. This approach ensures that the system model is always evaluated at *feasible discrete variable values*, and simultaneously seeks to retain the search characteristics of the basic PSO dynamics. Stagnation of particles owing to loss of population diversity has been one of the major drawbacks of the PSO algorithm; this undesirable search attribute becomes more pronounced in the case of *constrained single objective mixed-discrete* problems. A new efficient technique is developed to characterize the population diversity at a particular iteration. The estimated diversity measure is used to apply (i) a dynamic repulsion away from the best global solution for the continuous variables and (ii) a stochastic update of the discrete variables for each solution. This approach is uniquely helpful in preserving the population diversity in PSO. The generalized diversity measure formulated in this paper can be used for diversity preservation in a wide variety of population-based algorithms.

The Mixed-Discrete PSO (MDPSO) algorithm outperforms the normal PSO, when applied to a set of standard unconstrained problems involving objective functions that present different types of complexities: e.g., multimodality, high non-linearity, and extensive flat regions. The algorithm is also tested on a comprehensive set of ninety-eight MINLP problems. MDPSO performs well for the lower dimensional MINLP problems. Interestingly, MDPSO generally performs better for the purely integer test problems compared to the mixed-integer problems. Overall, MDPSO finds the point of minimum with a relative accuracy of 1e−03 or lower in around 50 % of the entire set of MINLP test runs. More problem specific assignment of the prescribed MDPSO parameter values is expected to help in obtaining more accurate solutions for such a wide variety of mixed-discrete optimization problems. When compared to a popular binary genetic algorithm, MDPSO performed better for the low dimensional MINLP problems, and provided comparable results for the higher dimensional MINLP problems. The MDPSO algorithm is also employed to perform wind farm design, where the selection of turbine-types to be installed (discrete) and the farm layout (continuous) are simultaneously optimized. A remarkable increase in the overall farm power generation (60 %) is accomplished, which illustrates the potential of applying the MDPSO algorithm to practical mixed-discrete engineering design problems. Future research in Mixed-Discrete PSO should focus on a comprehensive parametric analysis of the sensitivity of the algorithm performance to the prescribed parameter values. Subsequent development of standard guidelines to specify prescribed parameters values based on problem complexity would further enhance the universal applicability of this class of mixed-discrete optimization algorithms.

Acknowledgments Support from the National Science Foundation Awards CMMI-1100948, and CMMI-0946765 is gratefully acknowledged. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

References

Alatas B, Akin E, Ozer AB (2009) Chaos embedded particle swarm optimization algorithms. Chaos Solitons Fractals 40(4):1715–1734

Banks A, Vincent J, Anyakoha C (2007) A review of particle swarm optimization. Part I: background and development. Nat Comput 6(4):467–484

Banks A, Vincent J, Anyakoha C (2008) A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Nat Comput 7(1):109–124

Beyer HG, Lange B, Waldl HP (1996) Modelling tools for wind farm upgrading. In: European union wind energy conference. AIAA, Goborg, Sweden

Blackwell TM, Bentley P (2002) Don't push me! Collision avoiding swarms, vol 2. In: 2002 IEEE congress on evolutionary computation. IEEE Computer Society, Honolulu, HI

Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems, 1st edn. Oxford University Press, Cary, NC

Bussieck MR, Drud AS, Meeraus A (2007) Minlplib—a collection of test models for mixed-integer nonlinear programming. Tech Rep, GAMS Development Corp, Washington, DC 20007, USA

Cabrera JF, Coello CAC (2007) Handling constraints in particle swarm optimization using a small population size. In: 7th Mexican international conference on artificial intelligence: advances in artificial intelligence. SMIA, Aguascalientes, Mexico, pp 41–51

Chowdhury S, Dulikravich GS (2010) Improvements to single-objective constrained predator–prey evolutionary optimization algorithm. Struct Multidisc Optim 41(4):541–554

Chowdhury S, Dulikravich GS, Moral RJ (2009) Modified predator–prey algorithm for constrained and unconstrained multi-objective optimisation. Int J Math Model Numer Optim 1(1/2):1–38

Chowdhury S, Messac A, Khire R (2010a) Developing a non-gradient based mixed-discrete optimization approach for comprehensive product platform planning (cp^3). In: 13th AIAA/ISSMO multidisciplinary analysis optimization conference. AIAA, Fort Worth, TX

Chowdhury S, Zhang J, Messac A, Castillo L (2010b) Exploring key factors influencing optimal farm design using mixed-discrete particle swarm optimization. In: 13th AIAA/ISSMO multidisciplinary analysis optimization conference. AIAA, Fort Worth, TX

Chowdhury S, Messac A, Khire R (2011) Comprehensive product platform planning (cp^3) framework. ASME J Mech Des (special issue on Designing Complex Engineered Systems) 133(10):101004

Chowdhury S, Zhang J, Messac A, Castillo L (2012) Unrestricted wind farm layout optimization (UWFLO): investigating key factors influencing the maximum power generation. Renew Energy 38(1):16–30

CMU, IBM (2010) CMU-IBM Cyber-infrastructure for MINLP. http://www.minlp.org/index.php. Accessed 1 Dec 2010

Corne D, Dorigo M, Glover F (1999) New ideas in optimisation (Advanced topics in computer science). McGraw-Hill, New York City, NY

Deb K (2009) Multi-Objective optimization using evolutionary algorithms. Wiley, Chichester, UK

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: Nsga-II. IEEE Trans Evol Comput 6(2):182–197

GamsWorld (2010) MINLP World. http://www.gamsworld.org/minlp/index.htm. Accessed 1 Dec 2010

GE-Energy (2009) 1.5 mw wind turbine. http://www.ge-energy.com/products and services/products/wind turbines/index.jsp. Accessed 1 Dec 2009

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning, 1st edn. Addison-Wesley Professional, USA

Gonzalez JS, Rodriguez AGG, Mora JC, Santos JR, Payan MB (2010) Optimization of wind farm turbines layout using an evolutive algorithm. Renew Energy 35(8):1671–1681

Grady SA, Hussaini MY, Abdullah MM (2005) Placement of wind turbines using genetic algorithms. Renew Energy 30(2):259–270

Hu X, Eberhart RC (2002) Solving constrained nonlinear optimization problems with particle swarm optimization. In: Sixth world multiconference on systemics, cybernetics and informatics (SCI), Orlando, FL, USA

Jarboui B, Damak N, Siarry P, Rebai A (2008) A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Appl Math Comput 195:299–308

Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: IEEE international conference on neural networks. IEEE, Piscataway, NJ, USA, IV, pp 1942–1948

Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: IEEE international conference on in systems, man, and cybernetics, vol 5. IEEE, pp 4104–4108

Kennedy J, Eberhart RC, Shi Y (2001) Swarm Intelligence, 1st edn. Morgan Kaufmann, USA

Kitayama S, Arakawa M, Yamazaki K (2006) Penalty function approach for the mixed discrete nonlinear problems by particle swarm optimization. Struct Multidisc Optim 32:191–202

Krink T, Vesterstrom JS, Riget J (2002) Particle swarm optimisation with spatial particle extension. In: 2002 IEEE congress on evolutionary computation, vol 2. IEEE Computer Society, Honolulu, HI

Laskari EC, Parsopoulos KE, Vrahatis MN (2002) Particle swarm optimization for integer programming. In: IEEE 2002 congress on evolutionary computation. IEEE, Hawaii, USA

Miele A, Cantrell JW (1969) Study on a memory gradient method for the minimization of functions. J Optim Theory Appl 3(6):459–470

Mikkelsen R, Sorensen JN, Oye S, Troldborg N (2007) Analysis of power enhancement for a row of wind turbines using the actuator line technique. J Phys 75(1)

Murugan P, Kannan S, Baskar S (2009) Application of NSGA-II algorithm to single-objective transmission constrained generation expansion planning. IEEE Trans Power Syst 24(4):1790–1797

Nakamichi Y, Arita T (2004) Diversity control in ant colony optimization. Artif Life Robot 7(4):198–204

NDSU (2010) North Dakota agricultural weather network. http://ndawn.ndsu.nodak.edu/. Accessed 1 Dec 2010

Parsopoulos KE, Vrahatis MN (2002) Particle swarm method for constrained optimization problems. In: Euro-international symposium on computational intelligence, Kosice, Slovakia

Pohlheim H (2010) GEATbx: example functions (single and multi-objective functions) 2 parametric optimization

Ponsich A, Azzaro-Pantel C, Domenech S, Pibouleau L (2007) Mixed-integer nonlinear programming optimization strategies for batch plant design problems. Ind Eng Chem Res 46(3):854–863

Ponsich A, Azzaro-Pantel C, Domenech S, Pibouleau L (2008) Mixed-integer nonlinear programming optimization strategies for batch plant design problems. Chem Eng Process 47:420–434

Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchial particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 8(3):240–255

Riget J, Vesterstrom JS (2002) A diversity-guided particle swarm optimizer—the arpso. Tech Rep 2002-02, EVALife Project Group, Department of Computer Science, Aarhus Universitet

Schittkowski K (2009) A collection of 100 test problems for nonlinear mixed-integer programming in fortran: user's guide. Department of Computer Science, University of Bayreuth, Bayreuth, Germany

Shi Y, Eberhart RC (1998) Parameter selection in particle swarm optimization. In: Evolutionary programming VII. New York, USA, pp 591–600

Silva A, Neves A, Costa E (2002) An empirical comparison of particle swarm and predator prey optimization. In: 13th Irish International conference on artificial intelligence and cognitive science, Limerick, Ireland, vol 2464, pp 103–110

Singh G, Grandhi RV, Stargel DS (2010) Modified particle swarm optimization for a multimodal mixed-variable laser peening process. Struct Multidisc Optim 42:769–782

Sisbot S, Turgut O, Tunc M, Camdali U (2010) Optimal positioning of wind turbines on gokceada using multi-objective genetic algorithm. Wind Energy 13(4):297–306

Sobol M (1976) Uniformly distributed sequences with an additional uniform property. USSR Comput Math Math Phys 16: 236–242

Sorensen P, Nielsen T (2006) Recalibrating wind turbine wake model parameters—validating the wake model performance for large offshore wind farms. In: European wind energy conference and exhibition, EWEA, Athens, Greece

Tasgetiren MF, Suganthan PN, Pan QQ (2007) A discrete particle swarm optimization algorithm for the generalized traveling salesman problem. In: 9th annual conference on genetic and evolutionary computation (GECCO). SIGEVO and ACM, London, UK, pp 158–167

Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 85:317–325

Venter G, Haftka RT (2009) Constrained particle swarm optimization using a bi-objective formulation. Struct Multidisc Optim 40: 65–76

Wang Y, Cai Z, Zhou Y (2009) Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization. Int J Numer Methods Eng 77:1501–1534

Xie XF, Yang WJZZL (2002) A dissipative particle swarm optimization. In: 2002 IEEE congress on evolutionary computation, vol 2. IEEE Computer Society, Honolulu, HI, pp 1456–1461

Zavala AEM, Diharce ERV, Aguirre AH (2005) Particle evolutionary swarm for design reliability optimization, vol 3410. In: Evolutionary multi-criterion optimization: third international conference, EMO 2005. Springer, Guanajuato, Mexico, pp 856–869

Zhang W, Li H, Zhao Q, Wang H (2009) Guidelines for parameter selection in particle swarm optimization according to control theory. In: Fifth international conference on natural computation. IEEE, Tianjin, China

Zhang J, Chowdhury S, Messac A, Castillo L (2013) A multivariate and multimodal wind distribution model. Renewable Energy 51: 436–447