# A Hybrid Simplex Search and Particle Swarm Optimization for Unconstrained Optimization

Shu-Kai S. Fan[†] and Erwie Zahara

*Department of Industrial Engineering and Management*

*Yuan Ze University*

*Taoyuan County, Taiwan 320, Republic of China*

November 19, 2003

Corresponding Author: Shu-Kai S. Fan, Associate Professor
Department of Industrial Engineering & Management
Yuan Ze University
135 Far East Road, Chung-Li, Taoyuan County
320, Taiwan, ROC
Phone: +886-3-4638800 ext. 510
Fax: +886-3-4638907
E-mail: simonfan@saturn.yzu.edu.tw, morrisfan@r2r.iem.yzu.edu.tw

[†] To whom correspondence should be addressed. E-mail: simonfan@saturn.yzu.edu.tw

## Abstract

This paper proposes the hybrid NM-PSO algorithm based on the Nelder-Mead (NM) simplex search method and particle swarm optimization (PSO) for unconstrained optimization. NM-PSO is very easy to implement in practice since it does not require gradient computation. The modification of both the Nelder-Mead simplex search method and particle swarm optimization intends to produce faster and more accurate convergence. In a suite of 20 test function problems taken from the literature, computational results via an experimental study show that the hybrid NM-PSO approach outperforms the three other search techniques (i.e., the original NM simplex search method, the original PSO and the guaranteed convergence particle swarm optimization (GCPSO)) in terms of solution quality and convergence rate. The new algorithm is demonstrated to be extremely effective and efficient at locating best-practice optimal solutions.

## Scope and purpose

The function minimization techniques have been applied extensively in physical sciences, with applications in root finding of polynomials, system of equations, in estimating the parameters of nonlinear functions, and in searching the optimum parameter setting of scientific systems or engineering processes. The Nelder-Mead (NM) simplex search method is a very popular, efficient direct search method for function minimization without using derivatives. Particle swarm optimization (PSO) is one of the most recent evolutionary optimization methods developed for solving continuous optimization problems. The current study investigates the hybridization of the above two methods, and the performance of the hybrid algorithm is compared with other alternatives via simulations. The substantial improvements upon the effectiveness, efficiency and accuracy are reported to justify the claim that the hybrid approach presents an excellent tradeoff between exploitation in NM and exploration in PSO.

**Index Terms:** simplex search method, particle swarm optimization, unconstrained optimization.

## I. Introduction

Function minimization is employed extensively in physical sciences, with applications in root finding of polynomials, system of equations and in estimating the parameters of nonlinear functions [1]. The focus of this paper is thus on solving unconstrained minimization problems, and we present a hybrid method that tries to find a potential global minimum of a real-valued function $f(\boldsymbol{x})$ of N real-valued variables.

Nelder and Mead (NM) have proposed a simplex search method [2]-[4], which is a simple direct search technique that has been widely used in unconstrained optimization scenarios. One of the reasons for its popularity is that this method is easy to use and does not need the derivatives of the function under exploration. This is a very important feature in many applications where gradient information is not always available. However, one has to be very careful when using this method since it is very sensitive to the choice of initial points and not guaranteed to attain the global optimum.

Based upon the interaction of individual entities called "particles," particle swarm optimization (PSO) presented by Kennedy and Eberhart's [5]-[6] is an evolutionary computation technique in which each particle flies through the multi-dimensional search space with a velocity, which is constantly updated by the particle's best previous performance and by the best previous performance of the particle's neighbors. To date, PSO has been used successfully to optimize various continuous nonlinear functions.

Although PSO does eventually locate the desired solution, practical use of an evolutionary computation technique in solving complex optimization problems is severely limited by the high computational cost of the slow convergence rate [7]. The convergence rate of PSO is also typically slower than those of local direct search techniques (*e.g.*, Hooke and Jeeves method [8] and Nelder-Mead simplex search method), as they do not utilize much local information to determine a most promising search direction.

To deal with the slow convergence of PSO, an idea to combine PSO with a local simplex search technique is addressed in this paper; the rationale behind this is that such a hybrid approach expects to enjoy the merits of PSO with those of a local simplex search technique. In other words, PSO contributes to the hybrid approach in a way to ensure that the search is less likely to be trapped in local optima that often arise from employing a pure local search technique, while the local simplex search part of the hybrid approach makes the search converge faster than pure PSO. Roughly speaking, the hybrid approach can usually exploit a better tradeoff between computational efforts and global optimality of the solution found.

To verify the claim that the convergence rate and effectiveness of PSO can be greatly improved, a hybrid approach combining PSO with the Nelder-Mead simplex search method for unconstrained optimization is compared extensively with the following three optimization methods:

1) the simplex search method developed by Nelder-Mead [2];
2) the PSO method developed by Kennedy-Eberhart [6];
3) the modified PSO method introduced by van den Bergh [9].

The hybrid approach will be demonstrated via computational studies to be superior to the three optimization methods mentioned above, as will be discussed shortly in section IV.

The rest of the paper is outlined as follows. Section II reviews the fundamentals of simplex search method, PSO method and modified PSO method. Section III presents the NM-PSO hybrid structure and algorithm. Section IV illustrates efficiency and accuracy of the hybrid approach in solving unconstrained optimization problems through computational comparisons. Finally, major results of the paper are summarized in section V along with some remarks of areas for future research.

## II. Simplex Search Method and Particle Swarm Optimization

Optimization techniques can be classified into two broad categories: traditional direct search techniques (*e.g.*, the simplex search method, Rosenbrock's method [10], *etc*.) and heuristic techniques (*e.g.*, PSO, genetic algorithms, neural networks, *etc*.). Note that the methods of gradient-based types are beyond the scope of this research. The first major difference between NM and PSO is that the choice of initial points in the simplex search method is pre-determined but PSO initializes a swarm with a set of random points. The second difference is that PSO proceeds by attracting new points towards those points that have better function values, whereas the simplex search method evolves by moving away from a point that has the worst performance. The advantage of the simplex search method is that it is straightforward in an algorithmic sense and computationally efficient. However, as a result of using only local information, when they converge to a stationary point, there is no guarantee that the global optimum is found. In contrast, a PSO method explores the global search space without using local information of promising search directions. Consequently, it is less likely to be trapped in local optima, but its computational cost is comparably higher. In short, a PSO method often focuses on "exploration;" the simplex method focuses on "exploitation." Making most use of the characteristics of each method, it is reasonable to anticipate that the hybrid method would exhibit a mixture of exploitation and exploration properties nicely. As expected, the problem that the Nelder-Mead simplex search method is sensitive to initial conditions can be resolved by this hybridization. van den Bergh [9] also presented the trade-off analysis between exploration and exploitation that can be obtained or tuned by appropriate selection of inertia weight and acceleration constant values.

A. *The Nelder-Mead Simplex Search Method (NM)*

The Nelder-Mead simplex search method is based upon the work of Spendley *et al.* [11], which is a local search method designed for unconstrained optimization without using gradient information. The operations of this method are to rescale the simplex based on the local behavior of the function by using four basic procedures: reflection, expansion, contraction and shrinkage. Through these procedures, the simplex can successfully improve itself and get closer to the optimum. The original NM simplex procedure is outlined below and the pivot steps of the NM algorithm are illustrated in Figure 1 through a two-dimensional case ( $\mathrm{N}=2$ ).

1. **Initialization.** For the minimization of a function of N variables, create N+1 vertex points to form an initial N-dimensional simplex. Evaluate the functional value at each vertex point of the simplex. See a two-dimensional simplex exhibited in (a) of Figure 1. For the maximization case, it is convenient to transform the problem into the minimization case by pre-multiplying the objective function by –1.
2. **Reflection.** In each iteration, determine $P_{high}, P_{\sec hi}, P_{low}$ vertices, indicating vertex points that have the highest, the second highest, and the lowest function values, respectively. Let $f_{high}, f_{\sec hi}, f_{low}$ represent the corresponding observed function values. Find $P_{cent}$, the center of the simplex excluding $P_{high}$ in the minimization case. Generate a new vertex $P_{refl}$ by reflecting the worst point according to the following equation (see (a) of Figure 1):

$$P_{refl} = (1+\alpha)P_{cent} - \alpha P_{high} \tag{1}$$

where $\alpha$ is the reflection coefficient ($\alpha$ >0). Nelder and Mead suggested the use of $\alpha$ =1. If $f_{low} \leq f_{refl} \leq f_{\sec hi}$, accept the reflection by replacing $P_{high}$ with $P_{refl}$, and step 2 is entered again for a new iteration.

3. **Expansion.** Should reflection produce a function value smaller than $f_{low}$ (*i.e.*, $f_{refl} < f_{low}$), the reflection is expanded in order to extend the search space in the same direction and the expansion point is calculated by the following equation (see (b) of Figure 1):

$$P_{\exp} = \gamma P_{refl} + (1-\gamma)P_{cent} \tag{2}$$

where $\gamma$ is the expansion coefficient $(\gamma > 1)$. Nelder and Mead suggested $\gamma$ =2. If $f_{\exp} < f_{low}$, the expansion is accepted by replacing $P_{high}$ with $P_{\exp}$; otherwise, $P_{refl}$ replaces $P_{high}$. The algorithm continues with a new iteration in step 2.

4. **Contraction.** When $f_{refl} > f_{\sec hi}$ and $f_{refl} \leq f_{high}$, then $P_{refl}$ replaces $P_{high}$ and contraction is tried (see I of Figure 1). If $f_{refl} > f_{high}$, then direct contraction without the replacement of $P_{high}$ by $P_{refl}$ is performed (see (d) of Figure 1). The contraction vertex is calculated by the following equation:

$$P_{cont} = \beta P_{high} + (1-\beta)P_{cent} \tag{3}$$

where $\beta$ is the contraction coefficient $(0 < \beta < 1)$. Nelder and Mead suggested $\beta$ =0.5. If $f_{cont} \leq f_{high}$, the contraction is accepted by replacing $P_{high}$ with $P_{cont}$ and then a new iteration begins with step 2.

5. **Shrink.** If $f_{cont} > f_{high}$ in step 4, contraction has failed and shrinkage will be the next attempt. This is done by shrinking the entire simplex (except $P_{low}$) by (see (e) and (f) of Figure 1):

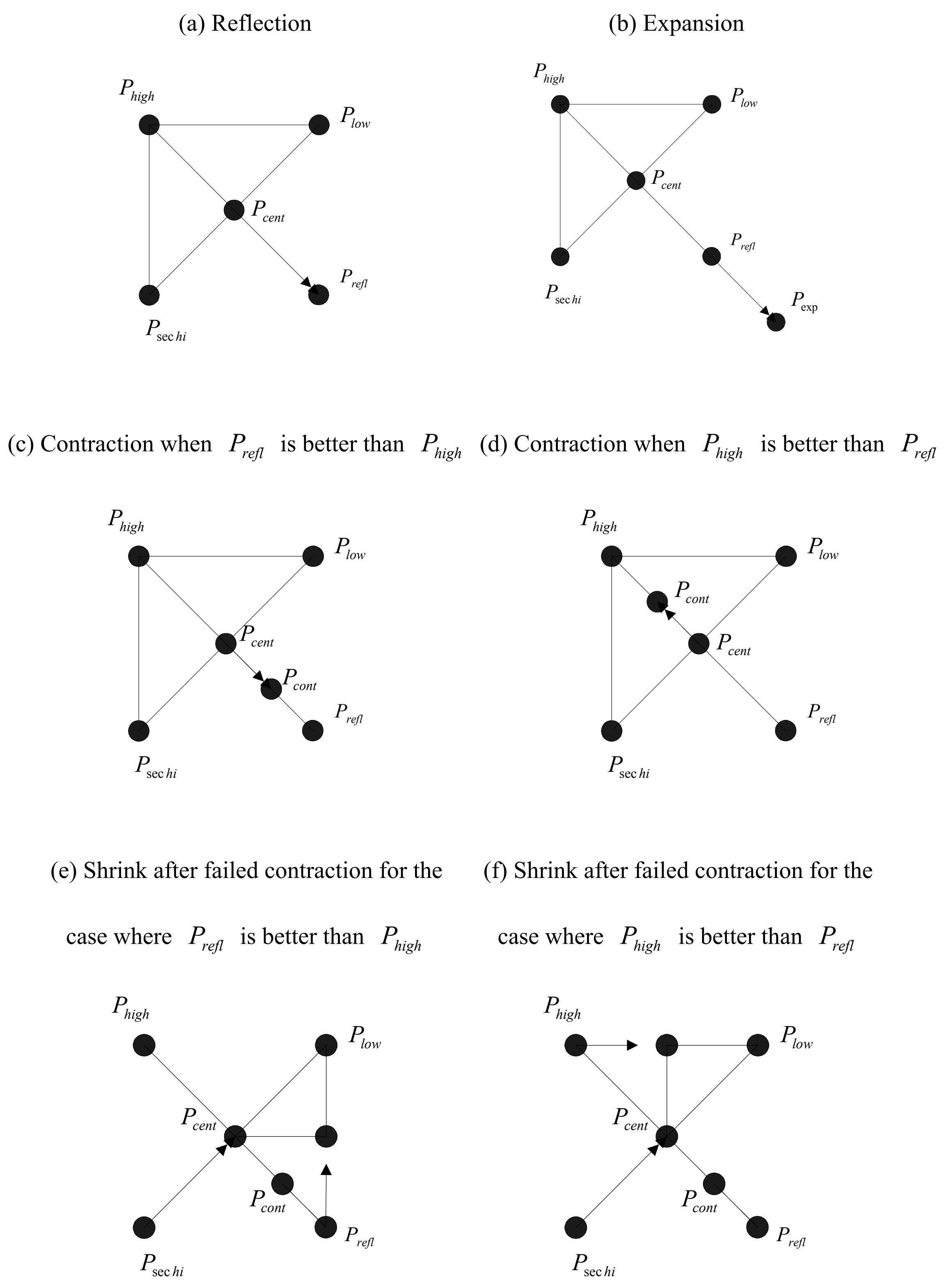

(a) Reflection
(b) Expansion
$P_{high}$
$P_{low}$
$P_{cent}$
$P_{refl}$
$P_{\sec hi}$
$P_{\exp}$
(c) Contraction when $P_{refl}$ is better than $P_{high}$
(d) Contraction when $P_{high}$ is better than $P_{refl}$
$P_{cont}$
(e) Shrink after failed contraction for the case where $P_{refl}$ is better than $P_{high}$
(f) Shrink after failed contraction for the case where $P_{high}$ is better than $P_{refl}$

Figure 1. Nelder-Mead pivot operations.

$$P_i \leftarrow \delta P_i + (1-\delta)P_{low} \tag{4}$$

where $\delta$ is the shrinkage coefficient $(0 < \delta < 1)$. Nelder and Mead suggested $\delta$=0.5. The algorithm then evaluates function values at each vertex (except $P_{low}$) and returns to step 2 to start a new iteration.

The Nelder-Mead simplex method has been applied in physics [12], crystallography [13], biology [14], chemistry [15] and health care [16]. Fletcher [17] considers the Nelder-Mead technique as one of the most successful methods that merely compare function values. There have been abundant studies devoted to various modifications of the Nelder-Mead simplex method appearing in the literature, such as Barton and Ivey [18], Nazareth and Tzeng [19], etc. However, the unmodified Nelder-Mead version of the simplex algorithm is best suited to our needs for comparison purposes.

*B. Particle Swarm Optimization (PSO)*

Before the introduction of the PSO procedure, previous optimization approaches have been developed based on mimicking the evolutionary process as frequently seen in nature. Just as survival of the fittest promotes the better of the entire population in the long run, the filtering operation (crossover and/or mutation) found in these approaches eventually leads to satisfactory solutions. PSO is also population-based and evolutionary in nature, with one major difference is that the PSO has memory in terms of the inertia weight, and then the social exchange information. Instead, a commonly observed social behavior, where members of a group tend to follow the lead of the best of the group, is simulated by PSO. As simple in concept and economic in terms of computational costs, PSO has a definite edge over other evolutionary optimization techniques. The procedure of PSO is reviewed below.

1. **Initialization.** Randomly generate a swarm of the potential solutions, called "particles" and assign a random velocity to each. The population size is problem-dependent and the one most commonly used in PSO is often between 20 and 50 [20]. For the computational experiments conducted in section IV, the population size of 5N particles is sampled from Uniform(-50,50) for solving N-dimensional problems for PSO.
2. **Velocity Update.** The particles are then "flown" through hyperspace by updating their own velocity. The velocity update of a particle is dynamically adjusted, subject to its own past flight and those of its companions. The particle's velocity and position are updated by the following equations:

$$\begin{aligned} V_{id}^{New}(t+1) = w \times V_{id}^{old}(t) + c_1 \times rand() \times (p_{id}(t) - x_{id}^{old}(t)) \\ + c_2 \times rand() \times (p_{gd}(t) - x_{id}^{old}(t)) \end{aligned} \tag{5}$$

$$x_{id}^{New}(t+1) = x_{id}^{old}(t) + V_{id}^{New}(t+1) \tag{6}$$

where $c_1$ and $c_2$ are two positive constants; $w$ is an inertia weight and rand( ) is a random value inside (0, 1). Eberhart and Shi [21] and Hu and Eberhart [22] suggested $c_1 = c_2 = 2$ and $w = [0.5 + (rand()/2.0)]$. Equation (5) illustrates the calculation of a new velocity for each individual. The velocity of each particle is updated according to its previous velocity ($V_{id}$), the particle's previous best location ($p_{id}$) and the global best location ($p_{gd}$). Particle's velocities on each dimension are clamped to a maximum velocity $V_{\max}$, where the maximum velocity $V_{\max}$ is a fraction of the domain search space in each dimension. Equation (6) shows how each particle's position is updated in the search space.

As far, PSO is one of the most recent evolutionary optimization methods. One of the reasons that PSO is attractive is that there are only a very few parameters that need to be adjusted. Although PSO is still in its infancy, it has been used across a wide range of applications, including end milling [23], reactive power and voltage control [24], mass-spring system [25] and neural network training [26, 27]. Generally speaking, PSO like other evolutionary optimization algorithms is applicable to most optimization problems and circumstances that can be cast into optimization problems.

*C. Guaranteed Convergence Particle Swarm Optimization (GCPSO)*

Recently, van den Bergh [9] and Clerc and Kennedy [28] have done extensive analysis of particle trajectories within the PSO to determine how to guarantee convergent swarm behavior. van den Bergh [9] found a dangerous property in the original PSO: if a particle's current position coincides with the global best position particle, the particle will only move away from this point if its previous velocity and $w$ are non-zero. If their previous velocities are very close to zero, then all particles will stop moving once they catch up with the global best particle, which may lead to premature convergence of the algorithm. To pre-actively counter this behavior in a particle swarm and to ensure convergence, van den Bergh and Engelbrecht [29] modified the original PSO method called the Guaranteed Convergence Particle Swarm Optimizer (GCPSO).

The GCPSO algorithm works as follows. Let $\tau$ be the index of the global best particle. The idea of GCPSO is then to update the position of the particle $\tau$ as:

$$x_{\tau d}^{New}(t+1) = p_{gd}(t) + wV_{\tau d}^{old}(t) + \rho(t)(1-2rand(\,)). \tag{7}$$

To achieve this, the velocity update of $\tau$ is defined as

$$V_{\tau d}^{New}(t+1) = w \times V_{\tau d}^{old}(t) + p_{gd}(t) - x_{\tau d}^{old}(t) + \rho(t)(1-2rand()). \tag{8}$$

In brief, the $-x_{\tau d}$ term "resets" the particle's position to the position $p_{gd}$, $wV_{id}^{old}$ signifies a search direction, and $\rho(t)(1-2rand(\,))$ term generates a random search term with side length $2\rho(t)$; $\rho(0)$ is initialized to 1.0, with $\rho(t)$ defined as

$$\rho(t+1) = \begin{cases} 2\rho(t) & \text{if } \#successes > s_c \\ 0.5\rho(t) & \text{if } \#failures > f_c \\ \rho(t) & \text{otherwise,} \end{cases} \tag{9}$$

where the terms $\#failures$ and $\#successes$ denote the number of consecutive failures or successes, respectively. A failure is defined as $f(x_{gd}(t)) \approx f(x_{gd}(t-1))$, indicating no considerable function improvement occurs between the two consecutive global best particles. The values $s_c$ and $f_c$ are threshold parameters and it is recommended to set $f_c = 5,\ s_c = 15.$ The following additional rules must also be implemented to ensure that equation (9) is well defined:

$$\#successes(t+1) > \#successes(t) \Rightarrow \#failures(t+1) = 0,$$

$$\#failures(t+1) > \#failures(t) \Rightarrow \#successes(t+1) = 0.$$

Thus, on a success the failure count is set to zero, and likewise the success count is reset when a failure occurs.

Note that only the best particle in the swarm uses the modified updates in equations (7) and (8), and the rest of the swarm uses the normal velocity and position updates defined in equations (5) and (6). For the computational experiments in Section IV, GPCSO uses a swarm size of 20 which is randomly generated with a Uniform(-50, 50) for solving N-dimensional problems. The stopping criterion adopted by GPCSO for solving problems of size higher than 10 dimensions contains 10000 iterations, and as to other test problems the stopping criterion will be discussed in section IV.

## III. Hybrid NM-PSO Method

The hybrid idea behind the methods introduced in section II is to combine their advantages and avoid disadvantages. Similar ideas have been discussed in hybrid methods using genetic algorithm and direct search technique. These hybrid techniques emphasized the tradeoff between accuracy, reliability and computation time in global optimization [30]-[31]. This section introduces the hybrid method and, in doing so, also demonstrates that the convergence of the simplex method and the accuracy of the PSO method can be further improved simultaneously.

## *A. The structure of our NM-PSO hybrid*

Figure 2 depicts the schematic representation of the proposed hybrid NM-PSO. When solving an N-dimensional problem, the hybrid approach takes 3N+1 particles. The initial swarm, N+1 particles, is constructed using the random generated starting point from Uniform(-50, 50) and a step size of 1.0 in each coordinate direction to form an initial simplex, and an additional 2 particles with opposite directions are randomly generated for each dimension (akin to the experimental exploration set out in EVOP [32]). For example, if the starting point is (0, 1) then the initial swarms for the NM and NM-PSO methods are illustrated in Figure 3. The additional swarm of 2N particles mentioned above may be a worthy investment as they could possibly bring about a great leap to the vicinity of the global optimum in the early iterations. A total of 3N+1 particles are sorted by fitness, and the best N particles are saved for subsequent use. The top N+1 particles in the figure 2 are fed into the modified simplex search method to improve the N+1$^{th}$ particle. Joined by the N best particles and the N+1$^{th}$ particle, the last 2N particles are adjusted by the modified PSO method (*i.e.*, selection, mutation for global best and velocity update). The result is sorted in preparation for repeating the entire run. The algorithm of this simplex-PSO approach is summarized in Figure 4 and the algorithm terminates when it satisfies a convergence criterion. The stopping criterion will be presented in Section IV.

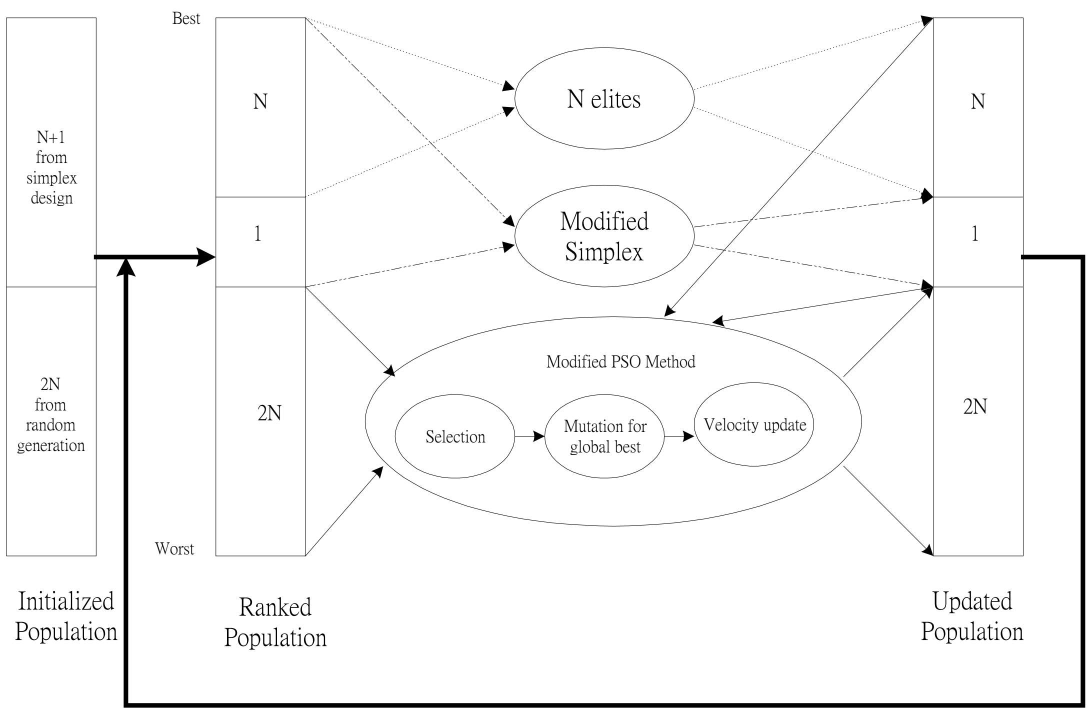


Figure 2. Schematic representation of the NM-PSO hybrid.

| | |
|---|---|
| ..............▶ | Representation of the selection on N elites particles |
| —·—·—·▶ | Representation of the modified simplex operation |
| ———▶ | Representation of the modified PSO operation |

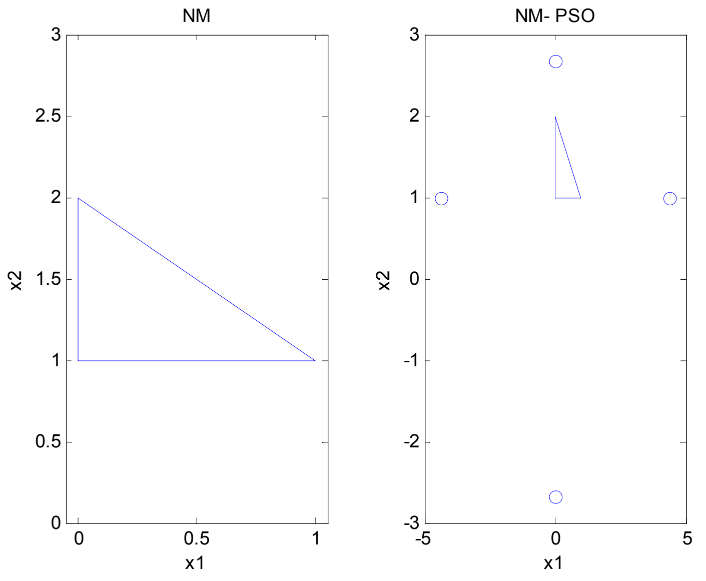


Figure 3. Initial populations for NM, PSO, NM-PSO for the starting point (0,1) in a two-dimensional case.

```
1. Initialization. Generate a population of size 3N+1.
Repeat
    2. Evaluation & Ranking. Evaluate the fitness of each particle.
       Rank them based on the fitness results.
    3. N Elites. Save the top N elites.
    4. Modified Simplex. Apply a simplex operator to the top N+1 particles and
       replace the N+1th particle with the update.
    5. Modified PSO. Apply modified PSO operator for updating 2N particles with
       worst fitness.
Until a termination criterion is reached.
```

Figure 4. The NM-PSO hybrid algorithm.

B. *The modified simplex search method*

If the optimal solution of an N-dimensional problem is very far away from the starting point, a second expansion operator may help improve the convergence rate. This operator will apply only after the success of an expansion attempt. In this situation, the current simplex might be very likely remote from the optimal solution. A detailed description of this second expansion is as follows:

**Second expansion.** If $f_{\exp} < f_{low}$ (after expansion has been performed), the expansion is performed again in order to extend the search space in the same promising direction and the second expansion point is calculated by the following equation:

$$P_{second\exp} = \theta P_{\exp} + (1-\theta)P_{cent} \qquad (10)$$

where $\theta$ is the second expansion coefficient $(\theta > 1)$. The choice of $\theta$=2 has been tested with much success from early computational experience. If $f_{second\exp} < f_{low}$, the second expansion is accepted by replacing $P_{high}$ with $P_{second\exp}$; otherwise, $P_{\exp}$ replaces $P_{high}$. A new iteration is started. Figure 5 summaries the modified simplex search method algorithm.

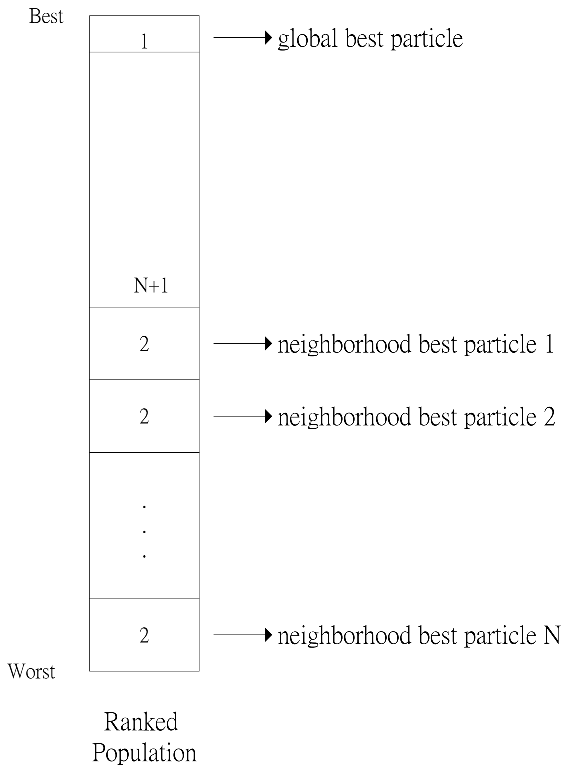
1. From ranked population, select N+1 particles.
2. Attempt reflection. If the reflection is accepted then attempt expansion, else attempt contraction or replace the worst fitness point with the reflection point.
3. Attempt expansion. If the expansion is accepted then attempt second expansion, else replace the worst fitness point with the expansion point.
4. Attempt second expansion. If the second expansion is accepted then replace the worst fitness point with the second expansion point, else replace the worst fitness point with the first expansion point.
5. Attempt contraction. If the contraction is accepted then replace the worst fitness point with the contraction point, else attempt shrinking for the entire simplex (except the best fitness point).

Figure 5. The modified simplex search algorithm.

## C. *The modified PSO method*

First, the modified PSO method begins with the selection of the global best particle ($p_{gd}$) and the neighborhood best particles ($p_{ld}$). The global best particle is selected from the entire population which has been sorted by fitness, and the N neighborhood best particles are selected from the worst 2N particles which are divided into N groups of two where the better of each group is selected. See the illustration in Figure 6. The particle's previous best location ( $p_{id}$ ) used for a velocity update as shown in equation (5) has been replaced by the neighborhood best particles position ( $p_{ld}$ ), and equation (5) becomes

$$\begin{aligned} V_{id}^{New}(t+1) = w \times V_{id}^{old}(t) + c_1 \times rand() \times (p_{ld}(t) - x_{id}^{old}(t)) \\ + c_2 \times rand() \times (p_{gd}(t) - x_{id}^{old}(t)) \end{aligned} \tag{11}$$



Best
1
global best particle
N+1
2
neighborhood best particle 1
2
neighborhood best particle 2
2
neighborhood best particle N
Worst
Ranked Population

Figure 6. The selection operator for modified PSO.

Looking at the original global-best PSO method, it is realized that the particles' velocity updates depend strongly on the global best particle. If the global best particle is trapped in a local optimum, then all the other particles will also fly toward the local optimum. For this situation, the velocity update for the global best particle though equations (5-6) generates merely a tiny jump for further improvement such that those particles are very unlikely to pull themselves out of the local optimum. To resolve this problem, a mutation heuristic is added to the global best particle as described below.

**Mutation heuristic for the global best particle**

Let $x^{old\_gbest}$ denote the position for the global best particle in an N-dimensional problem. At first, the heuristic uses normal distribution to randomly generate 5 particles based on $x^{old\_gbest}$ according to the following equation:

$$\mathbf{x}_i^{new\_gbest} = \mathbf{x}^{old\_gbest} + \boldsymbol{\varepsilon}, \qquad i=1, 2, 3, 4, 5. \tag{12}$$

where $\boldsymbol{\varepsilon}^T = [\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_N]$, $\varepsilon_j \sim N(0,\sigma)$, $j = 1, 2, \cdots, N$, and $\sigma$ is initially set to 1. If the ratio of the number of “successful” mutations to all 5 mutations is higher than 2/5, then $\sigma$ is increased to $(1/\lambda)\sigma$, else $\sigma$ is decreased to $\lambda\sigma$ for the next mutation, where $\lambda$ is the mutation coefficient $(\lambda < 1)$. The value of $\lambda = 0.85$ is opted herein and has been tested with success. If the ratio of the number of “successful” mutations to all mutations is 2/5, then $\sigma$ remains unchanged. Thus, the best fitness is culled from the five mutations to replace $x^{old\_gbest}$.

The mutation heuristic is an adaptation of PSO particularly to the instances of unconstrained optimization, in which case the maximum constrained velocity $V_{\max}$ need not be taken into account, therefore implying an acceleration of search speed. Higashi and Iba [33] had also presented particle swarm optimization with Gaussian mutation that combines the idea of the particle swarm with concepts from evolutionary algorithms. Their mutation scheme is similar to the one used in this study. The method selected particles at the predetermined probability and their positions are determined at the probability under the Gaussian distribution. Figure 7 summarizes the modified PSO algorithm for unconstrained optimization.

**Part I: Selection.** From the population select the global best particle and the neighborhood best particles.

**Part II: Mutation heuristic for the global best particle.** The algorithm is as below.

```
1. Define $x^{old\_gbest}$ as the global best particle, set the mutation success rate to 2/5
   and the mutation coefficient $(\lambda)$ to 0.85.
2. Generate 5 particles based on the position of the global best particle with
   variance $\sigma$ according to equation 11.
3. For each mutation
   {
       if mutation success rate = 2/5, then $\sigma^{new} = \sigma^{old}$.
       if mutation success rate > 2/5, then $\sigma^{new} = (1/\lambda)\sigma^{old}$
       if mutation success rate < 2/5, then $\sigma^{new} = \lambda\sigma^{old}$
   }
4. Replace the old global best particle ($x^{old\_gbest}$) with the new global best particle
   found among the $x_i^{new\_gbest}$.
```

**Part III: Velocity update.** Apply velocity update to the 2N particles with worst fitness according to:

$$V_{id}^{New}(t+1) = w \times V_{id}^{old}(t) + c_1 \times rand() \times (p_{ld}(t) - x_{id}^{old}(t)) + c_2 \times rand() \times (p_{gd}(t) - x_{id}^{old}(t))$$

$$x_{id}^{New}(t+1) = x_{id}^{old}(t) + V_{id}^{New}(t+1)$$

Figure 7. Algorithmic representation of the modified PSO.

## IV. Computational results

The ingredients of the hybrid NM-PSO method has been clearly described in section III. In this section, design of the experiments is explained, sensitivity analysis of NM-PSO parameters is explored and the empirical results are reported, which compare the hybrid approach with those of NM, PSO and GCPSO.

Table I. Test functions used for performance analysis

| Test function | Dim | Optimal OBJ value |
|---|---|---|
| 1. Powell badly scaled function | 2 | 0 |
| 2. B2 function | 2 | 0 |
| 3. Beale function | 2 | 0 |
| 4. Booth function | 2 | 0 |
| 5. Hellical valley function | 3 | 0 |
| 6. De Joung function | 3 | 0 |
| 7. Box three dimensional function | 3 | 0 |
| 8. Wood function | 4 | 0 |
| 9. Trigonometric function | 4 | 0 |
| 10. Extended Rosenbrock function | 4 | 0 |
| 11. Variably dimensioned function | 4 | 0 |
| 12. Penalty function Ⅰ | 8 | 5.42152..e-5 |
| 13. Penalty function Ⅱ | 8 | 1.23335..e-4 |
| 14. Trigonometric function | 8 | 0 |
| 15. Extended Powell function | 8 | 0 |
| 16. Griewank function | 10 | 0 |
| 17. Rastrigin function | 10 | 0 |
| 18. Extended Rosenbrock function | 10 | 0 |
| 19. Sphere function | 30 | 0 |
| 20. Griewank function | 50 | 0 |

A. *Design of the experiments*

Comparing the effectiveness of these algorithms requires large-scale testing on a variety of response functions. The NM-PSO will be tested against NM, PSO and GCPSO for a set of 20 deterministic test functions collected from Moré *et al*. [34] and van den Bergh [9]. They are a set of curvilinear functions for difficult unconstrained minimization problems. The forms and optimal values for the functions are described completely in the appendix. The variety of dimensions and functional forms make it possible to fairly assess the robustness (i.e., effectiveness, efficiency and accuracy) of the proposed approach within tolerably computational time. Many of these functions allow a choice of dimension, and an input dimension ranging from 2 to 50 for each test function is given in Table I.

Typically, there are three widely accepted stopping criteria employed in optimization methods. A stopping criterion used for this study is based on the simplex size over N+1 best points of the population. The stopping criterion proposed by Dennis and Woods [35], is defined as:

$$(1/\Delta)\max_i \left\| P_i - P_{low} \right\| \le \upsilon, \qquad \Delta = \max\left(1, \left\| P_{low} \right\|\right) \tag{13}$$

where the maximization is over all best points $i$'s (for $i = 1, 2, \cdots, N+1$) in the current simplex, and $\|\cdot\|$ denotes the Euclidean norm. The Dennis and Woods criterion is used in the computational test described below, with $\upsilon = 1 \times 10^{-4}$. In order to achieve quicker convergence, the algorithms will stop when either (13) is satisfied or the number of iterations reaches 1000N. Notice that the stopping criterion adopted by GPCSO for test problems 19 and 20 (of problem sizes 30 and 50) contains 10000 iterations (i.e., 200000 function evaluations). In GCPSO a fixed warm size of 20 particles is applied universally, so GCPSO cannot calculate the simplex size over N+1 best points of the swarm. According to our stopping criterion in equation (13), they need 31 and 51 particles, respectively.

The initial starting point for NM and NM-PSO is sampled from Uniform(-50, 50) and the initial swarm population for PSO and GCPSO is also randomly generated from Uniform(-50, 50). The optimization task of NM, PSO, GCPSO and NM-PSO on each test function was run 100 times. To evaluate the algorithm's efficiency and effectiveness, we adopted the following criteria, which had been observed from 100 minimizations per test function: the rate of successful minimizations, the average of the objective function evaluation numbers and the average error. These criteria are defined precisely below.

As long as either one of the termination criteria is first reached, the algorithms stop and return the coordinate of a located point, and the objective function value "$\mathrm{FOBJ}_{\mathrm{ALG}}$" (algorithm) at this point. We compared this result with the known analytical minimum "$\mathrm{FOBJ}_{\mathrm{ANAL}}$" and considered this result to be "successful" if the following inequality holds:

$$\left|\mathrm{FOBJ}_{\mathrm{ALG}} - \mathrm{FOBJ}_{\mathrm{ANAL}}\right| < 0.001 \tag{14}$$

The average of the objective function evaluation numbers is evaluated in relation to only the "successful minimizations". The average error is defined as the average of FOBJ gap between the best successful point found and the known global optimum in relation to only the "successful minimizations" achieved by the algorithm. The simulations were run on a Pentium IV 2.4G with memory capacity 256 MB, using Matlab 6.5.

## *B. Sensitivity Analysis of NM-PSO Parameters*

This section will investigate the performance of various NM-PSO parameters using several benchmark functions (i.e., Powell badly scaled function, Beale function, Hellical valley function, Box three dimensional function and Wood function) selected from Table I. The rate of successful minimization (%) is used as the criterion for setting parameter values. The experiment is conducted with the original coefficients shown in Table XI. Each time one of the NM-PSO parameters is varied in a certain interval to see which value within this internal will result in the best performance in term of rate of successful minimization.

Table II shows the sensitivity analysis of the reflection coefficient. The choice of interval $[0.5, 2.0]$ used in this analysis was motivated by the original Nelder-Mead simplex search procedure, where a reflection coefficient greater than 0 was suggested for general usage. From this table, it is found that a reflection coefficient setting at 1.5 returns the best rate of successful minimization.

Table II. Sensitivity analysis of reflection coefficient

| Reflection coefficient ($\alpha$) | Rate of successful minimization (%) | | | | |
|---|---|---|---|---|---|
| | Powell | Beale | Hellical | Box | Wood |
| G. 50 | 100 | 84 | 95 | 94 | 99 |
| G. 75 | 100 | 92 | 95 | 96 | 99 |
| G. 00 | 100 | 92 | 100 | 95 | 98 |
| G. 25 | 100 | 94 | 73 | 99 | 98 |
| G. 50* | 100 | 100 | 100 | 100 | 100 |
| G. 75 | 98 | 90 | 98 | 100 | 98 |
| 2.00 | 97 | 82 | 95 | 98 | 99 |

Table III. Sensitivity analysis of expansion coefficient

| Expansion coefficient ($\gamma$) | Rate of successful minimization (%) | | | | |
|---|---|---|---|---|---|
| | Powell | Beale | Hellical | Box | Wood |
| G. 50 | 100 | 82 | 96 | 100 | 99 |
| G. 75 | 100 | 93 | 94 | 100 | 100 |
| G. 00 | 100 | 94 | 97 | 100 | 99 |
| G. 25 | 100 | 96 | 84 | 94 | 99 |
| G. 50 | 100 | 88 | 99 | 100 | 100 |
| G. 75* | 100 | 100 | 100 | 100 | 99 |
| 3.00 | 100 | 96 | 98 | 99 | 99 |

Table IV. Sensitivity analysis of second expansion coefficient

| Second Expansion coefficient ($\theta$) | Rate of successful minimization (%) | | | | |
|---|---|---|---|---|---|
| | Powell | Beale | Hellical | Box | Wood |
| G. 50 | 99 | 93 | 90 | 99 | 100 |
| G. 75 | 99 | 92 | 95 | 99 | 100 |
| G. 00* | 100 | 100 | 100 | 100 | 100 |
| G. 25 | 100 | 80 | 98 | 100 | 98 |
| G. 50 | 100 | 100 | 43 | 100 | 98 |
| G. 75 | 100 | 93 | 89 | 99 | 99 |
| 3.00 | 100 | 97 | 87 | 100 | 100 |

Table V. Sensitivity analysis of contraction coefficient

| Contraction coefficient ($\beta$) | Rate of successful minimization (%) | | | | |
|---|---|---|---|---|---|
| | Powell | Beale | Hellical | Box | Wood |
| G. 25 | 100 | 86 | 91 | 97 | 89 |
| G. 50 | 100 | 95 | 69 | 97 | 100 |
| 0.75* | 100 | 100 | 99 | 100 | 99 |

Table VI. Sensitivity analysis of shrinking coefficient

| Shrinking coefficient ($\delta$) | Rate of successful minimization (%) | | | | |
|---|---|---|---|---|---|
| | Powell | Beale | Hellical | Box | Wood |
| G. 25 | 100 | 71 | 100 | 100 | 97 |
| G. 50* | 100 | 100 | 99 | 100 | 100 |
| 0.75 | 99 | 97 | 99 | 98 | 99 |

Table VII. Sensitivity analysis of inertia weight

| inertia weight $w$ | Rate of successful minimization (%) | | | | |
|---|---|---|---|---|---|
| | Powell | Beale | Hellical | Box | Wood |
| $0.5+(rand(\,)/2.0)*$ | 100 | 98 | 100 | 100 | 99 |
| *rand( )* | 100 | 100 | 91 | 100 | 100 |

Table VIII. Sensitivity analysis of $c_1$ coefficient

| $c_1$ coefficient | Rate of successful minimization (%) | | | | |
|---|---|---|---|---|---|
| | Powell | Beale | Hellical | Box | Wood |
| G. 20 | 99 | 96 | 96 | 100 | 99 |
| G. 40 | 99 | 96 | 95 | 100 | 100 |
| G. 60* | 100 | 100 | 100 | 100 | 100 |
| G. 80 | 100 | 89 | 90 | 99 | 100 |
| G. 00 | 100 | 98 | 90 | 99 | 100 |
| G. 20 | 100 | 89 | 93 | 99 | 98 |
| G. 40 | 100 | 97 | 94 | 99 | 100 |
| G. 60 | 100 | 97 | 97 | 97 | 100 |
| G. 80 | 100 | 98 | 100 | 100 | 100 |
| 2.00 | 100 | 82 | 93 | 95 | 99 |

Table IX. Sensitivity analysis of $c_2$ coefficient

| $c_2$ coefficient | Rate of successful minimization (%) | | | | |
|---|---|---|---|---|---|
| | Powell | Beale | Hellical | Box | Wood |
| G. 20 | 100 | 97 | 96 | 100 | 100 |
| G. 40 | 100 | 94 | 89 | 100 | 99 |
| G. 60 | 100 | 90 | 85 | 100 | 100 |
| G. 80 | 100 | 97 | 71 | 100 | 100 |
| G. 00 | 100 | 91 | 92 | 99 | 100 |
| G. 20 | 100 | 99 | 100 | 100 | 99 |
| G. 40 | 100 | 90 | 100 | 100 | 99 |
| G. 60* | 100 | 100 | 100 | 100 | 100 |
| G. 80 | 100 | 97 | 84 | 99 | 100 |
| 2.00 | 100 | 92 | 94 | 98 | 99 |

Table X. Sensitivity analysis of mutation coefficient

| Mutation coefficient ($\lambda$) | Rate of successful minimization (%) | | | | |
|---|---|---|---|---|---|
| | Powell | Beale | Hellical | Box | Wood |
| G. 25 | 100 | 93 | 93 | 100 | 99 |
| G. 40 | 99 | 100 | 88 | 100 | 99 |
| G. 55 | 100 | 100 | 98 | 99 | 99 |
| G. 70 | 99 | 96 | 99 | 100 | 100 |
| 0.85* | 100 | 98 | 100 | 100 | 98 |

Table XI. Best suggested for NM-PSO parameters after sensitivity analysis

| NM-PSO parameters | Original | Best-suggested |
|---|---|---|
| Reflection coefficient ($\alpha$) | G. 00 | G. 50 |
| Expansion coefficient ($\gamma$) | G. 00 | G. 75 |
| Contraction coefficient ($\beta$) | G. 50 | G. 75 |
| Shrinking coefficient ($\delta$) | G. 50 | G. 50 |
| Second expansion coefficient ($\theta$) | G. 00 | G. 00 |
| Inertia weight coefficient ($w$) | $0.5+(rand(\,)/2.0)$ | $0.5+(rand(\,)/2.0)$ |
| $C_1$ coefficient | G. 00 | G. 60 |
| $C_2$ coefficient | G. 00 | G. 60 |
| Mutation coefficient ($\lambda$) | 0.85 | 0.85 |

Table III to VI show the sensitivity analysis of expansion coefficient, second expansion coefficient, contraction coefficient and shrinking coefficient. The choice of their respective intervals has been motivated by the Nelder-Mead simplex search procedure. From Table III to VI found that expansion coefficient, second expansion coefficient, contraction coefficient and shrinking coefficient setting at 1.5, 2.0, 0.75 and 0.5 returns the best rate of successful minimization.

Table VII illustrates the sensitivity analysis of the inertia weight, and from this table, it is found that setting the inertia weight to $0.5+(rand(\,)/2.0)$ is better than *rand*( ) for achieving a better rate of successful minimization.

Table VIII and IX describe the sensitivity analyses of $c_1$ and $c_2$ coefficients; the choice of interval $[0.2, 2.0]$ used in these analyses was inspired by the original PSO algorithm, where these two coefficients with values greater than 0 and smaller or equal than 2 have been suggested for general usage. Table VIII indicates that a $c_1$ coefficient setting to 0.6 returns the best rate of successful minimization and Table IX shows that a $c_2$ coefficient setting to 1.6 returns the best rate of successful minimization.

Table X is on the sensitivity analysis of the mutation coefficient; the choice of interval $[0.2, 0.8]$ used in mutation is initially motivated by the "contraction" operation in the Nelder-Mead simplex search procedure, where a contraction coefficient greater than 0 and smaller than 1 was suggested for use. A mutation coefficient setting to 0.85 has been found to return the best rate of successful minimization. We summarize the above findings in Table XI, and apply these parameter values in the test functions for the purpose of comparison with results of other algorithms. Note again that the parameter selection procedure is performed in a one-factor-at-a-time manner. For each sensitivity analysis in this section, only one parameter is varied each time, and the remaining parameters are kept at the values suggested by the original NM and PSO algorithms. The interaction relation between parameters is assumed unimportant.

## G. *Empirical Results*

The empirical evaluations showed in this section will demonstrate that the proposed NM-PSO approach is highly effective at finding the global optima for the unconstrained optimization problems. Figure 8 illustrates the performances of all four approaches on the first test function to find the global optimum by plotting the best fitness versus the number of iterations for a single run. It can obviously be seen from the figure that the hybrid NM-PSO method converges more quickly than the other three methods (see the number of iterations). The NM-PSO method's fitness drops quickly (close to zero) at the second iteration attributed to adding those extra 2N particles in PSO, so the global solution is then swiftly achieved due to the second expansion strategy in NM. For this case, the NM-PSO method performs better than the other three methods from the aspects of computational efficiency. Figure 9 portrays the search paths of NM, PSO, GCPSO and NM-PSO on the Powell badly scaled function. From these tracks, it can be seen that the NM, PSO and GCPSO methods begin to approach a local optimum and then is rerouted to the global optimum, explaining why this method needs more iterations for convergence. The NM-PSO is already near the optimum with one single jump, emphasizing why the NM-PSO method performs best among these four methods for test function 1.

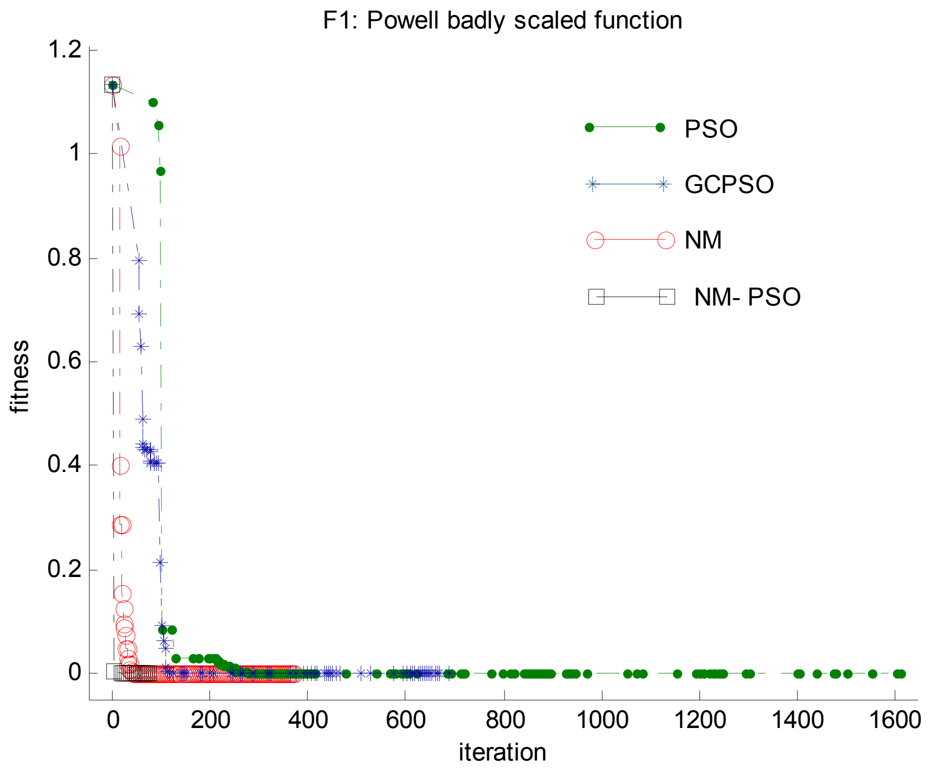


Figure 8. Best fitness of function 1 versus iteration for different approaches.

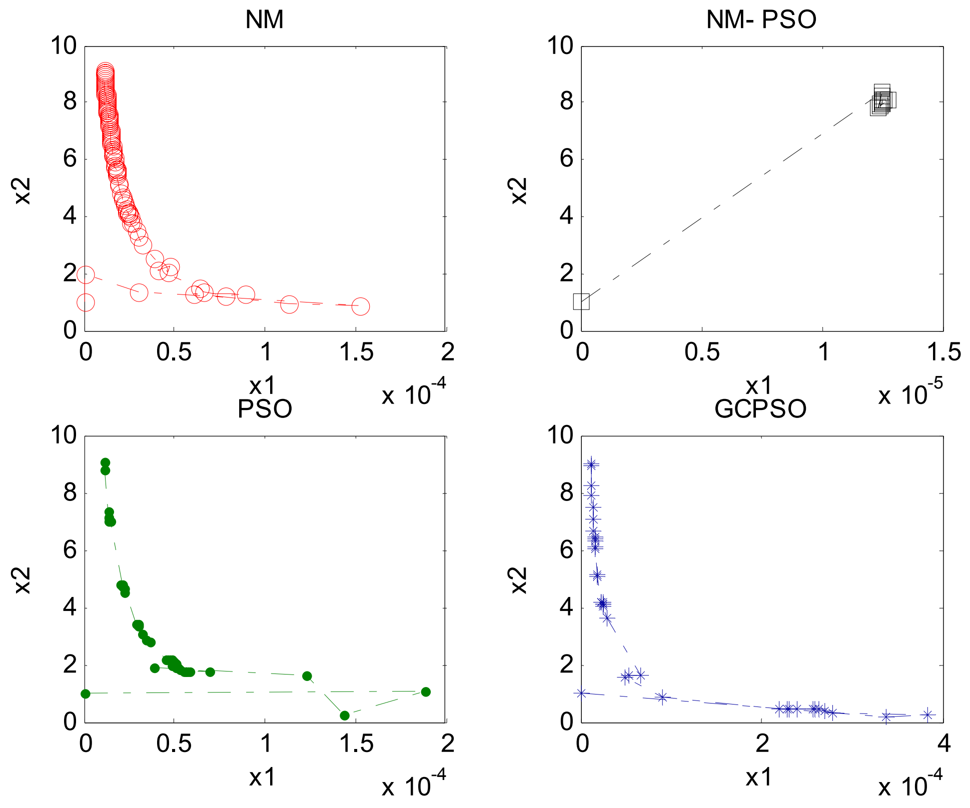


Figure 9. Solution paths of NM, PSO, GCPSO and NM-PSO on Powell badly scaled function. Note that the starting point is (0, 1) and the optimum point is $(1.098...10^{-5}, 9.106...)$.

Table XII shows a comparison of NM, PSO, GCPSO and NM-PSO results based on 20 test functions. Note again that the average of the objective function evaluation numbers and the average error in this table are evaluated in relation to only the "successful minimizations;" the numbers in parenthesis are the average of the objective function evaluation numbers and the average error over 100 runs. Comparing the simulation results of PSO and GCPSO in Table XII, they apparently exhibit the phenomenon described by van den Bergh [9]: the GCPSO algorithm has significantly higher rate of successful minimization and lower average of objective function evaluation numbers on unimodal functions than PSO. This improved performance is not visible on multi-modal functions (e.g., Rastrigin function and Grienwank function), because the GCPSO can still be trapped in local minima, just like the original PSO. Overall, the NM and PSO algorithms achieve about 60% success rate, 85% for the GCPSO algorithm and 96% for the hybrid NM-PSO. The averages of the objective function evaluation numbers of the NM-PSO algorithm for every test function are all significantly less than those of the PSO and GCPSO algorithms, indicating that the efficiency of the original PSO can be improved by the hybrid method. With respect to the statistics of average error, the NM-PSO algorithm prevails almost in every test function with an appreciable accuracy. Evidence obtained from the computational results in this section suggests that the hybrid approach provides an optimization solver with higher efficiency, reliability and accuracy for solving general unconstrained optimization problems. It is particularly important to note that the NM-PSO algorithm does not require gradient or Hessian matrix calculations, and therefore it does not suffer from the weakness of classical optimization methods, such as ill-conditioning.

## V. Conclusions

In this paper, a hybrid NM-PSO algorithm is presented for locating the global optima of continuous unconstrained optimization problems. The motivation of such a hybrid is to explore a better tradeoff between computational cost and global optimality of the solution attained. The initial population design of the NM-PSO method is an original idea in optimization methods and enables the hybrid mechanism to swiftly arrive at the neighborhood of the global optimum only after a few iterations elapsed. The second expansion strategy of the modified simplex search method makes it possible for a faster convergence rate, while the mutation heuristic strategy for the global best particle of the modified PSO method allows more latitude of search space to anchor the global optimum. Computational experience gained on a wide variety of test instances confirms the rich potential of the NM-PSO hybrid in solving deterministic unconstrained optimization problems. These observations also lead us to allege that the hybrid approach is indeed more accurate, reliable and efficient at locating best-practice optima than the other alternatives.

This hybrid NM-PSO is demonstrated to be a promising and viable tool to solve unconstrained nonlinear optimization problems. A number of improvements and extensions are currently being investigated by the authors. These include ways to accelerate the convergence for problems of higher dimension, as well as methods for extending the methodology to stochastic multi-objective systems. Practical applications of this hybrid approach in areas of classification, engineering process control, response surface optimization, and machine vision would also be worth further studying.

Table XII. A comparison of NM, PSO, GCPSO and NM-PSO results on 20 test functions

|  | Rate of successful minimization (%) |  |  |  | Average of objective function evaluation numbers |  |  |  | Average of FOBJ gaps between the best successful point found and the known global optimum |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No | NM | PSO | GCPSO | NM-PSO | NM | PSO | GCPSO | NM-PSO | NM | PSO | GCPSO | NM-PSO |
| 1 | 74 | 94 | 100 | 100 | 776<br>(631) | 20242<br>(20144) | 12375 | 2971 | 8.591e-6<br>(0.475) | 9.896e-6<br>692348675.92 | G.  668e-6 | G.  785e-6 |
| 2 | 78 | 100 | 100 | 100 | 159<br>(159) | 4188 | 2340 | 1124 | G.  727e-8<br>(0.099) | G.  460e-8 | G.  777e-8 | G.  235e-10 |
| 3 | 52 | 100 | 100 | 100 | 325<br>(2742) | 5440 | 2792 | 1458 | G.  575e-9<br>(0.845) | G.  986e-8 | G.  689e-8 | G.  607e-9 |
| 4 | 100 | 100 | 100 | 100 | 145 | 3848 | 2128 | 1065 | 2.622e-8 | 3.060e-8 | 9.722e-8 | 1.266e-9 |
| 5 | 46 | 99 | 96 | 100 | 380<br>(481) | 28736<br>(28620) | 6446<br>(6323) | 2552 | G.  389e-9<br>(6.257) | G.  183e-8<br>0.025 | G.  266e-7<br>(0.460) | G.  573e-9 |
| 6 | 100 | 100 | 100 | 100 | 283 | 9308 | 3031 | 1957 | 1.002e-9 | 8.806e-11 | 6.986e-10 | 1.630e-13 |
| 7 | 78 | 100 | 100 | 100 | 363<br>(329) | 44599 | 6432 | 3406 | G.  158e-5<br>1.595e+29 | G.  155e-15 | G.  448e-12 | G.  382e-11 |
| 8 | 99 | 13 | 79 | 100 | 1200<br>(1198) | 79930<br>(83488) | 34193<br>(33126) | 4769 | G.  882e-8<br>(0.079) | G.  445e-4<br>(0.050) | G.  386e-4<br>(0.001) | G.  714e-9 |
| 9 | 99 | 10 | 97 | 100 | 252<br>(251) | 84041<br>(84041) | 22615<br>(22457) | 71763 | G.  807e-4<br>(1.999e-4) | G.  919e-4<br>(0.333) | G.  897e-4<br>(2.633e-4) | G.  571e-5 |
| 10 | 100 | 49 | 100 | 100 | 682 | 84020<br>(84020) | 17170 | 3806 | G.  718e-7 | G.  487e-4<br>(3.259) | 8.686e-6 | G.  709e-9 |

Table XII (continued). A comparison of NM, PSO, GCPSO and NM-PSO results on 20 test functions

| | Rate of successful minimization (%) | | | | Average of objective function evaluation numbers | | | | Average of FOBJ gaps between the best successful point found and the known global optimum | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No | NM | PSO | GCPSO | NM-PSO | NM | PSO | GCPSO | NM-PSO | NM | PSO | GCPSO | NM-PSO |
| 11 | 100 | 100 | 100 | 100 | 639 | 41184 | 5591 | 4255 | 5.173e-9 | 8.446e-10 | 1.018e-8 | 1.344e-9 |
| 12 | 100 | 100 | 100 | 100 | 5796 | 328040 | 64040 | 44907 | 1.157e-5 | 2.941e-5 | 1.242e-11 | 1.306e-11 |
| 13 | 100 | 98 | 100 | 100 | 6034 | 328040 (328040) | 168020 | 114734 | G. 705e-6 | G. 648e-6 (9.861e-4) | G. 952e-8 | 8.955e-11 |
| 14 | 85 | 0 | 85 | 100 | 901 (901) | (328040) | 71152 (75469) | 255866 | G. 470e-5 (3.780e-4) | (10.909) | G. 967e-5 (1.637e-3) | G. 031e-5 |
| 15 | 66 | 89 | 100 | 100 | 6877 (5528) | 328040 (328040) | 128735 | 28239 | G. 830e-8 (4116.783) | G. 766e-4 (0.828) | G. 367e-7 | G. 134e-8 |
| 16 | 0 | 0 | 0 | 82 | (3270) | (504657) | (9253) | 14076 (13793) | (1.040) | (0.123) | (0.088) | G. 040e-11 (0.017) |
| 17 | 0 | 30 | 0 | 60 | (2887) | 510050 (509193) | (11146) | 12353 (12376) | (1164.238) | G. 080e-4 (1.021) | (7.771) | G. 911e-11 (4.836) |
| 18 | 11 | 0 | 100 | 100 | 10019 (6287) | (510050) | 176540 | 28836 | G. 429e-8 (375.058) | (1013.251) | G. 932e-4 | G. 378e-9 |
| 19 | 0 | 0 | 100 | 100 | (63552) | (4530150) | 200000 | 87004 | (726.704) | (4824.621) | G. 17e-16 | G. 763e-11 |
| 20 | 0 | 0 | 34 | 82 | (107857) | (12550250) | 200000 (200000) | 378354 (370682) | (1.230) | (6.575) | G. 469e-16 (0.028) | 9.969e-12 (0.021) |

## References

[1] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, New York, 1996.

[2] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *Computer Journal*, vol. 7, pp. 308-313, 1965.

[3] D. M. Olsson and L. S. Nelson, “ The Nelder-Mead simplex procedure for function minimization,” *Technometrics*, vol. 17, pp. 45-51, 1975.

[4] D. H. Chen, Z. Saleem and D. W. Grace, “ A new simplex procedure for function minimization,” *International Journal of Modeling and Simulation*, vol. 6, pp. 81-85, 1986.

[5] R. C. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” *Proc. Of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp. 39-43, 1995.

[6] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” *Proc. IEEE Int’l Conf. On Neural Networks*, Piscataway, NJ, USA, pp. 1942-1948, 1995.

[7] S. Smith, “The simplex method and evolutionary algorithms,” *Proc. IEEE Int’l Conf. On Evolutionary Computation*, pp. 799-804, 1998.

[8] R. Hooke and T. A. Jeeves, “Direct search solution of numerical and statistical problems,” *Journal of Association for Computing Machinery*, vol. 8, pp. 212-221, 1961.

[9] F. van den Bergh, *An Analysis of Particle Swarm Optimizers*, PhD Dissertation, University of Pretoria, Pretoria, 2001.

[10] H. H. Rosenbrock, “An automatic method for finding the greatest or least value of a function,” *Computer Journal*, vol. 3, pp. 175-184, 1960.

[11] W. Spendley, G. R. Hext and F. R. Himsworth, “Sequential application of simplex designs in optimization and evolutionary operation,” *Technometrics*, vol. 4, pp. 441-461, 1962.

[12] J. Balakrishnan, M. K. Gunasekaran and E. S. R. Gopal, “Critical dialectric – constant measurements in the binary liquid system – methanol + normal heptane,” *International Journal of PA Physics,* vol. 22, pp. 286-298, 1984.

[13] W. R. Busing and M. Matsui, “The application of external forces to computational model of crystals,” *Act Cryst* A, vol. 40, pp. 532-540, 1984.

[14] W. Schulze and U. Rehder, “Organization and morphogenesis of the human seminiferous epithelium,” *Cellular Tissue Review,* vol. 237, pp. 395-417, 1984.

[15] G. L. Silver, “Space Modification: An alternative approach to chemistry problem involving geometry,” *Journal of Computational Chemistry,* vol. 2, pp. 478-490, 1981.

[16] P. R. Sthapit, J. M. Ottoway, and G. S. Fell, “Determination of lead in matural and tap waters by flame atomic-fluorescence spectrometry,” *Analyst,* vol. 109, pp.1061-1075, 1984.

[17] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, Chichester, 1987.

[18] R. R. Barton and J. S. Ivey, “Nelder-Mead simplex modifications for simulation optimization,” *Management Science*, vol. 42, pp. 954-973, 1996.

[19] L. Nazareth and P. Tzeng, “Gilding the lily: a variant of the Nelder-Mead algorithm based on golden-section search,” *Computational Optimization and Applications*, vol. 22, pp. 133-134, 2002.

[20] X. Hu and R. C. Eberhart, “Adaptive particle swarm optimization: detection and response to dynamic systems,” *Proc. IEEE Int’l Conf. On Evolutionary Computation*, Honolulu, Hawaii, USA, pp. 1666-1670, 2002.

[21] R. C. Eberhart and Y. Shi, “Tracking and optimizing dynamic systems with particle swarms,” *Proc. Congress on Evolutionary Computation,* Seoul, Korea, pp. 94-97, 2001.

[22] X. Hu and R. C. Eberhart, “Tracking dynamic systems with PSO: where’s the cheese?” *Proc. of The Workshop on Particle Swarm Optimization,* Indianapolis, IN, USA, 2001.

[23] V. Tandon, “Closing the gap between CAD/CAM and optimized CNC and milling,” *Master’s thesis, Purdue School of Engineering and Technology*, Indianapolis, IN, USA, 2000.

[24] H. Yoshida, K. Kawata, Y. Fukuyama and Y. Nakanishi, “A particle swarm optimization for reactive power and voltage control considering voltage stability,” *Proc. Intl. Conf. on Intelligent System Application to Power Systems*, Rio de Janeiro, Brazil, pp. 117-121, 1999.

[25] B. Brandstatter and U. Baumgartner, “Particle swarm optimization—mass-spring system analogon,” *IEEE Transactions on Magnetics*, vol. 38, pp. 997-1000, 2002.

[26] R. C. Eberhart and X. Hu, “Human tremoe analysis using particle swarm optimization,” *Proc. of the congress on Evolutionary Comptation*, pp. 1927-1930, Washington D. C., USA, 1999.

[27] F. van den Bergh and A. P. Engelbrecht, “Cooperative learning in neural networks using particle swarm optimizers,” *South African Computer Journal*, vol. 26, pp. 84-90, 2000.

[28] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58-73, 2002.

[29] F. van den Bergh and A. P. Engelbrecht, "A new locally convergent particle swarm optimizer," *IEEE Intl. Conf. on Systems, Man and Cybernatics*, pp. 96-101, 2002.

[30] J. M. Renders and S. P. Flasse, "Hybrid methods using genetic algorithms for global optimization," *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 26, pp. 243-258, 1996.

[31] J. Yen, J. C. Liao, B. Lee and D. Randolph, "A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method," *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 28, pp. 173-191, 1998.

[32] G. E. P. Box, "Evolutionary operation: a method for increasing industrial productivity," *Applied Statistics*, vol. 6, pp. 81-101, 1957.

[33] N. Higashi and H. Iba, "Particle swarm optimization with Gaussian mutation," *IEEE Proc. on Swarm Intelligence Symposium,* pp. 72-79, Indianapolis, USA, 2003.

[34] J. J. Moré, B. S. Garbow and K. E. Hillstrom, "Testing unconstrained optimization software," *ACM Transactions on Mathematical Software*, vol. 7, pp. 17-41, 1981.

[35] J. E. Dennis, Jr. and D. J. Woods, "Optimization on microcomputers: the Nelder-Mead simplex algorithm," in *New Computing Environments: Microcomputers in Large Scale Computing,* A. Wouk (ED.), SIAM, Philadelphia, PA, pp. 116-122, 1987.

## Appendix

The 20 test functions we employed are given below. To define the test functions, we have adopted the following general format:

(Dimension): Name of function

(a) Function definition

(b) Global optimum

**Function 1 (2-D):** Powell badly scaled function

(a) $f(x) = (10x_1x_2 - 1)^2 + (\exp[-x_1] + \exp[-x_2] - 1.0001)^2$

(b) Global optimum with $f = 0$ at $(1.098...10^{-5}, 9.106...)$.

**Function 2 (2-D):** B2 function

(a) $f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$

(b) Global optimum with $f = 0$ at $(0, 0)$.

**Function 3 (2-D):** Beale function

(a) $f(x) = \sum_{i=1}^{3}\left(y_i - x_1(1 - x_2^i)\right)^2$, where $y_1 = 1.5,\ y_2 = 2.25,\ y_3 = 2.625$

(b) Global optimum with $f = 0$ at (3, 0.5).

**Function 4 (2-D):** Booth function

(a) $f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$

(b) Global optimum with $f = 0$ at (1, 3).

**Function 5 (3-D):** Helical valley function

(a) $f(x) = (10[x_3 - 10\theta(x_1, x_2)])^2 + (10[(x_1^2 + x_2^2)^{1/2} - 1])^2 + (x_3)^2$,

$$\text{where} \quad \theta(x_1, x_2) = \begin{cases} \dfrac{1}{2\pi}\arctan\left(\dfrac{x_2}{x_1}\right), & \textit{if } x_1 > 0 \\ \dfrac{1}{2\pi}\arctan\left(\dfrac{x_2}{x_1}\right), & \textit{if } x_1 < 0 \end{cases}$$

(b) Global optimum with $f = 0$ at (1, 0, 0).

**Function 6 (3-D):** De Joung function

(a) $f(x) = x_1^2 + x_2^2 + x_3^2$

(b) Global optimum with $f = 0$ at (0, 0, 0).

**Function 7 (3-D):** Box three-dimensional function

(a) $f(x) = \sum_{i=1}^{3} \left( \exp[-t_i x_1] - \exp[-t_i x_2] - x_3 (\exp[-t_i] - \exp[-10t_i]) \right)^2$

where $t_i = (0.1)i$

(b) Global optimum with $f = 0$ at (1, 10, 1), (10, 1, -1) and wherever $(x_1 = x_2 \ \ and \ \ x_3 = 0)$

**Function 8 (4-D):** Wood function

(a) $$\begin{aligned} f(x) &= \left(10(x_2 - x_1^2)\right)^2 + \left(1 - x_1\right)^2 + \left((90)^{1/2}(x_4 - x_3^2)\right)^2 + \left(1 - x_3\right)^2 \\ &+ \left((10)^{1/2}(x_2 + x_4 - 2)\right)^2 + \left((10)^{-1/2}(x_2 - x_4)\right)^2 \end{aligned}$$

(b) Global optimum with $f = 0$ at (1, 1, 1, 1).

**Function 9 (4-D):** Trigonometric function

(a) $$f_i(x) = 4 - \sum_{j=1}^{4} \cos x_j + i(1 - \cos x_i) - \sin x_i \qquad i = 1,2,....,4$$
$$f(x) = \sum_{j=1}^{4} f_j^2(x)$$

(b) Global optimum with $f = 0$.

**Function 10 (4-D):** Rosenbrock function

(a) $f(x) = \sum_{i=1}^{2} \left( 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2 \right)$

(b) Global optimum with $f = 0$ at (1, 1, 1, 1).

**Function 11 (4-D):** Variably dimensioned function

(a) $f(x) = \sum_{i=1}^{4} \left(x_i - 1\right)^2 + \left( \sum_{i=1}^{4} i(x_i - 1) \right)^2 + \left( \sum_{i=1}^{4} i(x_i - 1) \right)^4$

(b) Global optimum with $f = 0$ at (1, 1, 1, 1).

**Function 12 (8-D):** Penalty function I

(a) $f(x)=\sum_{i=1}^{8}\left((10^{-5})^{1/2}(x_i-1)\right)^2+\left(\left(\sum_{j=1}^{8}x_j^2\right)-\frac{1}{4}\right)^2$

(b) Global optimum with $f=5.42152...\times 10^{-5}$.

**Function 13 (8-D):** Penalty function II

$$f_1(x)=x_1-0.2$$

$$f_i(x)=a^{1/2}\left(\exp\left[\frac{x_i}{10}\right]+\exp\left[\frac{x_{i-1}}{10}\right]-y_i\right),\qquad 2\le i\le 8$$

(a) $$f_i(x)=a^{1/2}\left(\exp\left[\frac{x_{i-n+1}}{10}\right]-\exp\left[\frac{-1}{10}\right]\right),\qquad 8<i<16$$

$$f_{16}(x)=\left(\sum_{j=1}^{8}(8-j+1)x_j^2\right)-1$$

$$f(x)=\sum_{j=1}^{16}f_j^2(x)$$

where $a=10^{-5}$ and $y_i=\exp\left[\frac{i}{10}\right]+\exp\left[\frac{i-1}{10}\right]$

(b) Global optimum with $f=1.23335...\times 10^{-4}$.

**Function 14 (8-D):** Trigonometric function

(a)
$$f_i(x)=8-\sum_{j=1}^{8}\cos x_j+i(1-\cos x_i)-\sin x_i\qquad i=1,2,....,8$$
$$f(x)=\sum_{j=1}^{8}f_j^2(x)$$

(c) Global optimum with $f=0$.

**Function 15 (8-D):** Extended Powell function

(a)
$$f_{4i-3}(x)=x_{4i-3}+10x_{4i-2},\qquad i=1,2$$
$$f_{4i-2}(x)=5^{1/2}(x_{4i-1}-x_{4i}),\qquad i=1,2$$
$$f_{4i-1}(x)=(x_{4i-2}-2x_{4i-1})^2,\qquad i=1,2$$
$$f_{4i}(x)=10^{1/2}(x_{4i-3}-x_{4i})^2,\qquad i=1,2$$
$$f(x)=\sum_{j=1}^{8}f_j^2(x)$$

(b) Global optimum with $f=0$ at the origin.

**Function 16 (10-D):** Griewank function

(a) $\mathrm{f}(x) = \sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

(b) Global optimum with $f = 0$ at the origin.

**Function 17 (10-D):** Rastrigin function

(a) $f(x) = \sum_{i=1}^{10} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$

(c) Global optimum with $f = 0$ at origin.

**Function 18 (10-D):** Rosenbrock function

(a) $f(x) = \sum_{i=1}^{5} \left(100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2\right)$

(b) Global optimum with $f = 0$ at (1, 1, 1, 1).

**Function 19 (30-D):** Sphere function

(a) $f(x) = \sum_{i=1}^{30} x_i^2$

(b) Global optimum with $f = 0$ at the origin.

**Function 20 (50-D):** Griewank function

(a) $\mathrm{f}(x) = \sum_{i=1}^{50} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

(c) Global optimum with $f = 0$ at the origin.