DOI: 10.20472/IAC.2015.018.046

ZEINAB M. H. HENDAWY
Menofia University, Egypt

M. A. EL-SHORBAGY
Menofia university, Egypt

COMBINED TRUST REGION WITH PARTICLE SWARM FOR MULTI-OBJECTIVE OPTIMISATION

Abstract:

A novel approach is presented to solve multi-objective optimisation problems (MOOP).The algorithm combines the Trust Region (TR) algorithm with the Particle Swarm Optimisation (PSO) method.The MOOP is converted to a single objective optimisation problem (SOOP) using weighted method and some of the points in the search space are generated. For each point, the TR algorithm is used to solve the SOOP to obtain a point on the Pareto frontier. All points obtained are used as particle position for PSO to get all the points on the Pareto frontier. The algorithm is tested using several bench mark problems and coded using MATLAB 7.2 which show successful result in finding a Pareto optimal set.

Keywords:

Multi-objective Optimisation- Trust Region Method- Particle Swarm Optimisation- Weighted Method

INTRODUCTION

TR method generate steps with the help of a quadratic model of the objective function, define a region around the current iterate within which they trust the model to be an adequate representation of the objective function and then choose the step to be approximate minimzer of the model in this region. If a step is not acceptable, they reduce the size of the region and find a new minimize. In general, the direction of the step changes whenever the size of the TR is altered (Ou, 2011). To see the idea of TR, consider the unconstrained optimization problem:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{1}$$

where, f(x) is a nonlinear continuous differentiable function in $\mathbb{R}^n$. For a known iterate x_k the TR method determines subsequent iterate using:

$$x_{k+1} = x_k + d_k \tag{2}$$

where, d_k is trial step determined by minimizing a local quadratic (approximating) model of f at x_k (TR sub-problem) given by:

$$\begin{aligned} &\text{minimize} \quad q_k(d) = f_k + \nabla f_k^T d + \frac{1}{2} d^T H_k d \\ &\text{subject to} \quad \|d\| \le \Delta_k, \end{aligned} \tag{3}$$

where, H_k is Hessian of f(x) or approximate to it and $\Delta_k>0$ is the TR radius. Using the ratio:

$$r_k = \frac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)}, \tag{4}$$

traditional TR methods evaluate an agreement between the model and the objective function. The trial step d_k is accepted whenever r_k is greater than a positive constant. This leads us to the new point $x_{k+1} = x_k + d_k$ and the TR radius is updated. Otherwise, the TR radius must be diminished and the sub-problem (3) must be solved again (Ahookhosh *et al*., 2012).

Because of the boundedness of the TR, TR algorithms can use non-convex approximate models. This is one of the advantages of TR algorithms comparing with line search algorithms. TR algorithms are reliable and robust, they can be applied to ill-conditioned problems, they have very strong convergence properties and have been proven to be theoretically and practically effective and efficient for unconstrained and equality constrained optimization problems (Ahookhosh and Amini, 2010; Zhang *et al*., 2010). Also, The TR algorithm has proven to be a very successful globalization technique for nonlinear programming problems with equality and inequality constraints (El-Sobky, 2012)

For MOOPs, Kim and Ryu (2011) developed an iterative algorithm for bi-objective stochastic optimization problems based on the TR method and investigated different sampling schemes. Their algorithm does not require any strong modeling assumptions and has great potential to work well in various real-world settings.

PSO is an Evolutionary Computational (EC) model which is based on swarm intelligence. PSO is developed by Kennedy *et al*. (2001) who have been inspired by the research of the artificial livings. Similar to EC techniques, PSO is also an optimizer based on population. The system is initialized firstly in a set of randomly generated potential solutions and then performs the search for the optimum one iteratively. Whereas the PSO does not possess the crossover and mutation processes used in EC, it finds the optimum solution by swarms following the best particle. Compared to EC, the PSO has much more profound intelligent background and could be performed more easily. Based on its advantages, the PSO is not only suitable for

science research, but also engineering applications, in the fields of evolutionary computing, optimization and many others.

Multi-Objective Optimization (MOO) has been one of the most studied application areas of PSO algorithms. Number of approaches have been utilized and/or designed to tackle MOOPs using PSO. A straight forward approach is to convert MOO to a SOOP. Parsopoulos and Vrahatis (2002) presented a first study of the performance of the PSO in MOOPs. In recent years, many particle swarm algorithms were proposed for solving MOOPs (Mousa *et al*., 2012; Figueira *et al*., 2010). On the other hand, a comprehensive survey of the state-of-the-art in Multi-Objective (MO) particle swarm optimizers can be found in (Sierra and Coello, 2006) where different techniques reported in Multi-Objective Particle Swarm Optimization (MOPSO) development have been categorized and discussed.

This study presents a hybrid algorithm combining TR and PSO for solving MOOPs, which can overcome the disadvantage of the TR method (such as restrictions on the TR radius) and solve a class of MOOPs efficiently. It is a new algorithm that performs random searching and deterministic searching for solving MOOPs. In the proposed algorithm, first MOOP converting to SOOP, TR is used to obtain a point on the Pareto frontier and finally homogeneous PSO is applied to get all the points on the Pareto frontier.

MULTI-OBJECTIVE OPTIMIZATION

The MOO is a very important research area in engineering studies because real world design problems require the optimization of a group of objectives. Thanks to the effort of scientists and engineers during the last two decades, particularly the last decade, a wealth of MO optimizers have been developed and some MOOPs that could not be solved hitherto were successfully solved by using these optimizers (Tang, 2013). The general minimization problem of q objectives can be mathematically stated as:

$$\left.\begin{array}{ll} \text{minimize:} & f(x)=\left[f_j(x),\ j=1,2,...,q\right] \\ \text{subject to the constraints:} & Ci(x)\le 0,\quad i=1,2,...,p, \\ & Ce(x)=0,\quad e=1,2,...,m, \end{array}\right\} \quad (5)$$

where, $f_i(x)$ is the *j*-th objective function , $C_i(x)$ is the *i*-th inequality constraint, $C_e(x)$ is the e-th equality constraint and $x = [x_1,x_2,\ldots,x_n]$ is the vector of optimization or decision variables; where n the dimension of the decision variable space. The MOO problem then reduces to finding an *x* such that $f_j(x)$ is optimized. Since the notion of an optimum solution in MOOP is different compared to the SOOP, the concept of Pareto dominance is used for the evaluation of the solutions. This concept formulated by Vilfredo Pareto is defined as follows:

Definition 1. (Dominance Criteria). For a problem having more than one objective function (say, f_j j = 1,…,q, p>1), any two solution x_a and x_b can have one of two possibilities, one dominates the other or none dominates the other. A solution x_a is said to dominate the other solution x_b, if both the following condition are true. The solution x_a is no worse (say the operator p denotes worse and f denotes better) than x_b in all objectives, or $f_j(x_a) \not p f_j(x_b)$ for all j =1,..,q objectives.

The solution x_a is strictly better than x_b in at least one objective, or $f_j(x_a) f f_j(x_b)$ for at least one $j \in \{1,\ldots,q\}$.

If any of the above condition is violated, the solution x_a dose not dominates the solution x_b.

Definition 2. (Pareto optimal solution). X* is said to be a Pareto optimal solution of MOOP if there exists no other feasible x such that, $f_j(x) \le f_j(x^*)$ for all j = 1,…,q and $f_j(x) < f_j(x^*)$ for at least one objective function f_j.

WEIGHTED METHOD

Weighted method is an intuitive way for MOO. In this approach, different objectives are weighted and summed up to one single objective. By using weighted method (Friedrich *et al*., 2013), we convert the constrained MOOP (5) to SOOP. This method consists of creating a single-objective model by weighing the *q* objective functions by assigning a weight to each the functions. Through the weighted method the MOOP (5) is formulated as:

$$\begin{aligned} &\text{minimize} && f(x) = \sum_{j=1}^{q} w_j f_j(x) \\ &\text{subject to} && Ci(x) \le 0, \quad i = 1,2,...,p, \\ & && Ce(x) = 0, \quad e = 1,2,...,m, \end{aligned} \tag{6}$$

where, $w_1,\ldots,w_q$ are non-negative weights with $w_1+w_1+\ldots+w_q = 1$. The weights $w_1,\ldots,w_q$ are determined as follows:

$$w_j = random_j \Big/ \sum_{j=1}^{q} random_j, \forall\, j = 1,...,q \tag{7}$$

where, $random_1$, $random_2,\ldots$, $random_q$, are non-negative random integers. The following conclusions can be drawn for the weighted method:

- It is computationally very efficient
- It is conceptually very easy to understand
- Only one solution can be obtained in one run, assuming that the Pareto front is convex
- The solutions located in the concave region of the Pareto front cannot be obtained

PARTICLE SWARM OPTIMIZATION

PSO is an evolutionary computation technique motivated by the simulation of social behavior (Kennedy *et al*., 2001). Namely, each individual (agent) utilizes two important kinds of information in decision process. The first one is their own experience; that is, they have tried the choices and know which state has been better so far and they know how good it was. The second one is other agent's experiences; that is, they have knowledge of how the other agents around them have performed. Namely, they know which choices their neighbors have found are most positive so far and how positive the best pattern of choices was. In the PSO system, each agent makes his decision according to his own experiences and other agent's experiences. The system initially has a population of random solutions. Each potential solution, called a particle (agent), is given a random velocity and is flown through the problem space. The agents have memory and each agent keeps track of its previous (local) best position (called the P_{best}) and its corresponding fitness. There exist a number of P_{best} for the respective agents in the swarm and the agent with greatest fitness is called the global best (G_{best}) of the swarm. Each particle is treated as a point in a n-dimensional space. The i-th particle is represented as $x_i = (x_{i1},x_{i2},\ldots,x_{in})$. The best previous position of the i-th particle (P_{besti}) that gives the best fitness value is represented as $P_i = (p_{i1}, p_{i2},\ldots,p_{in})$. The best particle among all the particles in the population is represented by $P_g = (p_{g1}, p_{g2},\ldots,p_{gn})$. The velocity, i.e., the rate of the position change for particle i is represented as $v_i = (v_{i1},v_{i2},\ldots,v_{in})$.

The particles are manipulated according to the following equations (the superscripts denote the iteration):

$$v_i^{t+1} = w v_i^t + c_1 r_1 \left(p_i - x_i^t\right) + c_2 r_2 \left(p_g - x_i^t\right) \tag{8}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (9)$$

where, i = 1,2,….,N and N is the size of the population; w is the inertia weight; c_1 and c_2 are two positive constants, called the cognitive and social parameter respectively; r_1 and r_2 are random numbers uniformly distributed with in the range [0,1]. Equation 8 is used to determine the *i-th* particle's new velocity v_i^{t+1}, at each iteration *t*, while Equation 9 provides the new position of the *i-th* particle x_i^{t+1}, adding its newvelocity v_i^{t+1}, to its current position x_i^t. **Figure 1** shows the Description of velocity and position updates of a particle for a two-dimensional parameter space. **Figure 2** shows the pseudo code of the general PSO algorithm.

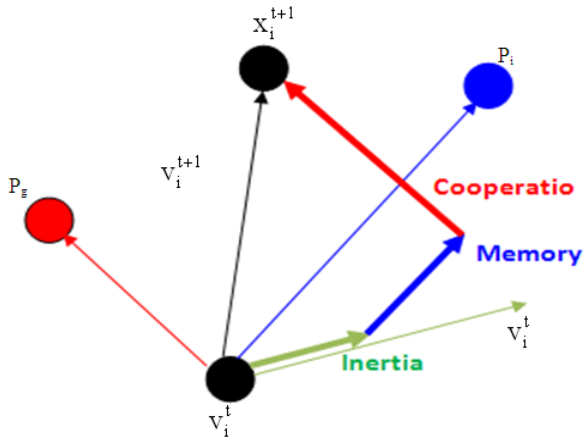

Fig. 1. Description of velocity and position updates in particle swarm optimization for a two dimensional parameter space

Randomly initialize positions and velocities of all particles.
Do:
  Set P_{best} and G_{best}.
  Calculate particle velocity according to equation (8).
  Update particle position according to equation (9).
  Evaluate the objective function value (fitness value).
while a satisfactory solution has been found

Fig. 2. The pseudo code of the general PSO algorithm

THE PROPOSED APPROACH

In the following, the proposed algorithm is presented. The proposed algorithm contains three stages initialization stage, TR stage and PSO stage.

Initialization stage

- Initialization
- Initialize N points in the search space

- Converting MOOP to SOOP
- The non-negative weights ($w_1,\ldots,w_m$) is generated using Equation 7
- Construct the weighted problem (6)
- Converting the general nonlinear optimization problem (6) to equality Constrained problem
- Following Dennis *et al*. (1999), we define the indicator matrix $W(x)\in \square^{p\times p}$, whose diagonal entries are

$$w_i(x)=\begin{cases}1 & \text{if } Ci(x)\geq 0\\ 0 & \text{if } Ci(x)<0\end{cases} \qquad (10)$$

Using this matrix, the Problem defined in Equation 6 can be transformed to the following equality constrained optimization problem:

$$\begin{aligned}&\text{minimize} && f(x)\\ &\text{subject to} && 1/2\,Ci(x)^T W(x)Ci(x)=0,\\ & && Ce(x)=0.\end{aligned} \qquad (11)$$

The above problem can be rewritten as:

$$\begin{aligned}&\text{minimize} && f(x)\\ &\text{subject to} && h(x)=0,\end{aligned} \qquad (12)$$

where, $h(x) = [C_e(x)\ 1/2C_i(x)^T W(x)C_i(x)]$.

The matrix W(x) is discontinuous; however, the function $W(x)C_i(x)$ is Lipschitz continuous and the function $C_i(x)^T W(x)C_i(x)$ is continuously differentiable (Dennis *et al*., 1999).

The Lagrangian function associated with problem defined in (12) is given by:

$$L(x_k,\lambda_k)=f(x_k)+\lambda_k^T h(x_k) \qquad (13)$$

where, $\lambda_k\in \square$ is the Lagrange multiplier vector associated with equality constraint $h(x_k)\in \square$.

The augmented Lagrangian is the function:

$$\Phi(x,\lambda;r)=L(x,\lambda)+r\left\|h(x_k)\right\|^2 \qquad (14)$$

where r> 0 is a parameter usually called the penalty parameter.

TR Stage

The detailed description of TR algorithm for solving problem (12) is presented.

The reduced Hessian approach is used to compute a trial step d_k. In this approach, the trial step d_k is decomposed into two orthogonal components; the normal component d_k^n and the tangential component d_k^t . The trial step d_k has the form $d_k=d_k^n+Z_k\bar{d}_k^t$, where Z_k is a matrix whose columns form an orthonormal basis for the null space of $\nabla h(x_k)^T$.

We obtain the normal component d_k^n by solving the following TR sub-problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\left\|h(x_k)+\nabla h(x_k)^T d^n\right\|^2 \\ &\text{subject to} && \left\|d^n\right\| \le \xi\Delta_k, \end{aligned} \qquad (15)$$

For some $\zeta \in (0,1)$.

Given the normal component d_k^n, we compute the tangential component $d_k^t = Z_k \bar{d}_k^t$ by solving the following TR sub-problem:

$$\begin{aligned} &\text{minimize} && \left[Z_k^T\left(\nabla_x L(x_k,\lambda_k)+H_k d_k^n\right)\right]^T \\ & && \bar{d}^t + \frac{1}{2}\bar{d}^{tT} Z_k^T H_k Z_k \bar{d}^t \\ &\text{subject to} && \left\|Z_k\bar{d}^t\right\| \le \sqrt{\Delta_k^2 - \left\|d_k^n\right\|^2}, \end{aligned} \qquad (16)$$

Once the trial step is computed, it needs to be tested to determine whether it will be accepted or not. To do that, a merit function is needed. We use the augmented Lagrangian function (14) as a merit function. To test the step, we compare the actual reduction in the merit function in moving from x_k to x_k+d_k versus the predicted reduction.

The actual reduction in the merit function is defined as:

$$\begin{aligned} &\text{Ared}_k = L(x_k,\lambda_k) - L(x_{k+1},\lambda_{k+1}) + r_k \\ &\left[\left\|h(x_k)\right\|^2 - \left\|h(x_{k+1})\right\|^2\right] \end{aligned} \qquad (17)$$

The predicted reduction in the merit function is defined as:

$$\begin{aligned} &\text{Pred}_k = -\nabla_x L(x_k,\lambda_k)^T d_k - \frac{1}{2} d_k^T H_k d_k - \\ &\Delta\lambda_k^T\left(h(x_k)+\nabla h(x_k)^T d_k\right) \\ &+ r_k\left[\left\|h(x_k)\right\|^2 - \left\|h(x_k)+\nabla h(x_k)^T d_k\right\|^2\right] \end{aligned} \qquad (18)$$

where, $\Delta\lambda_k = (\lambda_{k+1}-\lambda_k)$

If $(\text{Ared}_k/\text{Pred}_k)<\tau_0$ where $\tau_0 \in (0, 1)$ is a small fixed constant, then the step is rejected. In this case, the radius of the TR Δ_k is decreased by setting $\Delta_k = \tau_3 \|d_k\|$, where $\tau_3 \in (0, 1)$ and another trial step is computed using the new TR radius. If $(\text{Ared}_k/\text{Pred}_k) \ge \tau_2$, where $\tau_2>0$, then the step is accepted and set the TR as $\Delta_{k+1}=\min\{\Delta_{max},\text{Max}\{\Delta_{min},\tau_1\Delta_k\}\}$. If $\tau_0 \le (\text{A red}_k/\text{Pred}_k)<\tau_2$ then the step is accepted and set the TR as $\Delta_{k+1} = \max(\Delta_k,\Delta_{min})$. Finally, the algorithm is terminated when either $\|d_k\| \le \varepsilon_1$ or $\left\|Z_k^T\nabla_x L_k\right\| + \left\|h_k\right\| \le \varepsilon_2$, for some $\varepsilon_1,\varepsilon_2>0$. The pseudo code of TR stage showing in **Fig. 3**.

PSO stage

In this stage a homogeneous PSO for MOOP is proposed with a dynamic constriction factor (Abd-El-Wahed *et al*., 2011) to restrict velocity of the particles and control it. In homogeneous PSO one global repository concept is proposed for choosing pbest and *gbest*, this means that each particle has lost its own identity and treated simply as a member of social group. The procedure of the PSO stage is as follows.

Step1: Initialization

All non-dominated points (which obtained by applying TR stage) chosen as particles position x_i^t.

PSO parameters such as velocity v_i^t, inertia weight *w* and learning rates c_1 and c_2 are set up.

Store non-dominated particles in Pareto repository. If the specific constraint doesn't exist for a repository, the size of the repository is unlimited.

Step2: Evaluation

Evaluate the MO fitness value of each particle and save it in a vector form.

Step3: Floating

Two optimal solutions are chosen randomly for *pbest* and *gbest* from the repository.

Determine the new position of each particle with Equation 8 and 9.

Step4: Repairing of Particles:

where, the particle *i* start at the position x_i^t with velocity v_i^t in the feasible space, the new position x_i^{t+1} in **Fig. 3** depends on velocity v_i^{t+1}.

Choose $\varepsilon_1,\ \varepsilon_2,\ \tau_0,\ \tau_1,\ \tau_2,\ \tau_3,\ \ \Delta_0, \Delta_{\max}, \Delta_{\min}$ such that $\varepsilon_1 > 0$, $\varepsilon_2 > 0$, $0 < \tau_3 < 1 < \tau_1$, $0 \le \tau_0 \le \tau_2 < 1$, $\tau_2 > 0$, and $\Delta_{\min} \le \Delta_0 \le \Delta_{\max}$

For each point N $\left(x_0 \in \square^n\right)$, compute W_0, $H_0 \in \square^{n \times n}$, and set $k = 0$.

If $\left\|Z_k^T \nabla_x L_k\right\| + \left\|h_k\right\| \le \varepsilon_2 \rightarrow$ end for

Solve the sub-problem (15) to give the normal component d_k^n

Solve the sub-problem (16) to give the tangential component $d_k^t = Z_k \bar{d}_k^t$

Compute the trial step $d_k = d_k^n + Z_k \bar{d}_k^t$

If $\left\|d_k\right\| \le \varepsilon_2 \rightarrow$ end for

Compute $Ared_k$ and $Pred_k$

While $\left(Ared_k / Pred_k\right) < \tau_0 \rightarrow \Delta_k = \tau_3 \left\|d_k\right\| \rightarrow$ compute a new trial step d_k

If $\tau_0 \le \left(Ared_k / Pred_k\right) < \tau_2$, then $x_{k+1} = x_k + d_k \rightarrow \Delta_{k+1} = \max\left(\Delta_k, \Delta_{\min}\right)$.

Else $\left(Ared_k / Pred_k\right) \ge \tau_2$, then $x_{k+1} = x_k + d_k \rightarrow \Delta_{k+1} = \min\left\{\Delta_{\max},\ \max\left\{\Delta_{\min}, \tau_1 \Delta_k\right\}\right\}$.

Update H_{k+1}, W_{k+1}, and set $k = k + 1$

Fig. 3. The pseudo code of TR stage

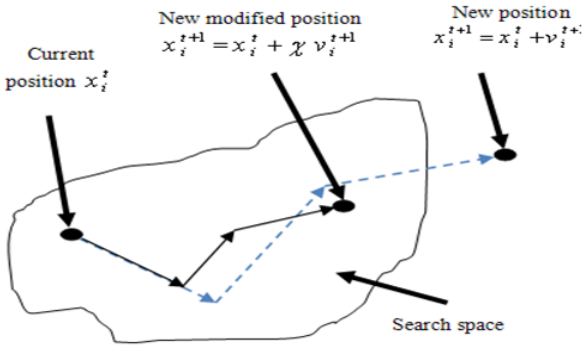

Fig. 4. The movement of the particle i through search space

Store non-dominated solution in Pareto repository

Chose non-dominated solution as position of particles x_i^t.

Initialize parameters for PSO $\left(v_i^t,\, w,\, c_1,\, c_2\right)$.

While (number of iterations, or the stopping criterion is not met)

Chosen randomly pbest and gbest from the repository.

Update particles velocity v_i^{t+1} and position x_i^{t+1} according to equation (8) and equation (9) of all particles.

Repair the unfeasible particle according to equation (20).

Evaluate fitness of particle swarm

Selection and update the repository

End while

Fig. 5. The pseudo code of PSO stage

To restrict (control) the particle's velocity $\mathrm{v_i^t}$, a modified constriction factor (i.e., dynamic constriction factor) is presented to keep the feasibility of the particles. E.g., **Fig. 4** shows the movement of the particle *i* through the search space without any control factor (dashed line) also with a modified constriction factor (solid line). Where the particle i start at position $\mathrm{x_i^t}$ with velocity $\mathrm{v_i^t}$ in the feasible space, the new position $\mathrm{x_i^{t+1}}$ depends on velocity $\mathrm{v_i^{t+1}}$ making the particle lose its feasibility, so we introduce a modified constriction factor:

$$\chi = \frac{2}{\left|-2-\tau-\sqrt{\tau^2+\tau}\right|} \qquad (19)$$

where, τ is the age of the infeasible particle (i.e., how long it is still infeasible) and it is increased with the number of failed trials to keep the feasibility of the particle. The new modified positions of the particles are computed as:

$$\mathrm{x_i^{t+1}} = \mathrm{x_i^t} + \chi\, \mathrm{v_i^{t+1}} \qquad (20)$$

For each particle, the feasibility is checked, if it is infeasible, the χ parameter is implemented to control its position and velocity.

Step5: Selection and Update the Repository

Check the Pareto optimality of each particle. If the fitness value of the particle is non-dominated when it compared to the Pareto optimal set in a repository, save it into the Pareto repository.

In the Pareto repository, if a particle is dominated from new one, then discard it.

Step6: Repeat

Repeat again step 2 to step 5 until the number of generation reaches to given τ.

The PSO stage algorithm needs at least two Pareto solutions in the first generation to avoid premature convergence. The pseudo code of PSO stage showing in **Fig. 5**. **Figure 6** shows the flow chart of proposed algorithm.

NUMERICAL RESULTS

In order to validate the proposed algorithm, several benchmark problems are solved which are reported in the literature (Deb, 2001). The algorithm is coded in MATLAB 7.2 and the simulations are run on a Pentium 4 CPU 900 MHz with 512 MB memory capacity. The parameters adopted in the implementation of the proposed algorithm are listed in **Table 1**.

Test Problems

For evaluating the performance of the proposed approach nine well-known MO benchmark problems are used. Each test problem consists of two objective functions with/without constraints and has continuous/discrete with convex/nonconvex Pareto front. The following test problems for study are considered (Deb, 2001):

Test Problem-1 (Continuous Convex):

$$\text{Minimize} \quad f_1(x) = x_1^2/4$$
$$\text{Minimize} \quad f_2(x) = x_1(1-x_2)+5$$

Subject to:

$$x_1 \in [0,10]$$
$$x_2 \in [0,10]$$

Test Problem-2 (Continuous Convex):

$$\text{Maximize} \quad f_1(x) = 1.1 - x_1$$
$$\text{Maximize} \quad f_2(x) = 60 - \frac{1+x_2}{x_1}$$

Subject to:

$$x_1 \in [0.1,1]$$
$$x_2 \in [0,5]$$

Test Problem-3 (Discrete):

$$\text{Minimize} \quad f_1(x) = x_1$$
$$\text{Minimize} \quad f_2(x) = g(x)h(x)$$

$$\text{where} \quad g(x) = 1 + 10x_2$$

$$\text{and} \quad h(x) = 1 - \left(\frac{f_1}{g}\right)^2 - \frac{f_1}{g}\sin(8\pi f_1)$$

Subject to:

$$x_1 \in [0,1]$$
$$x_2 \in [0,1]$$

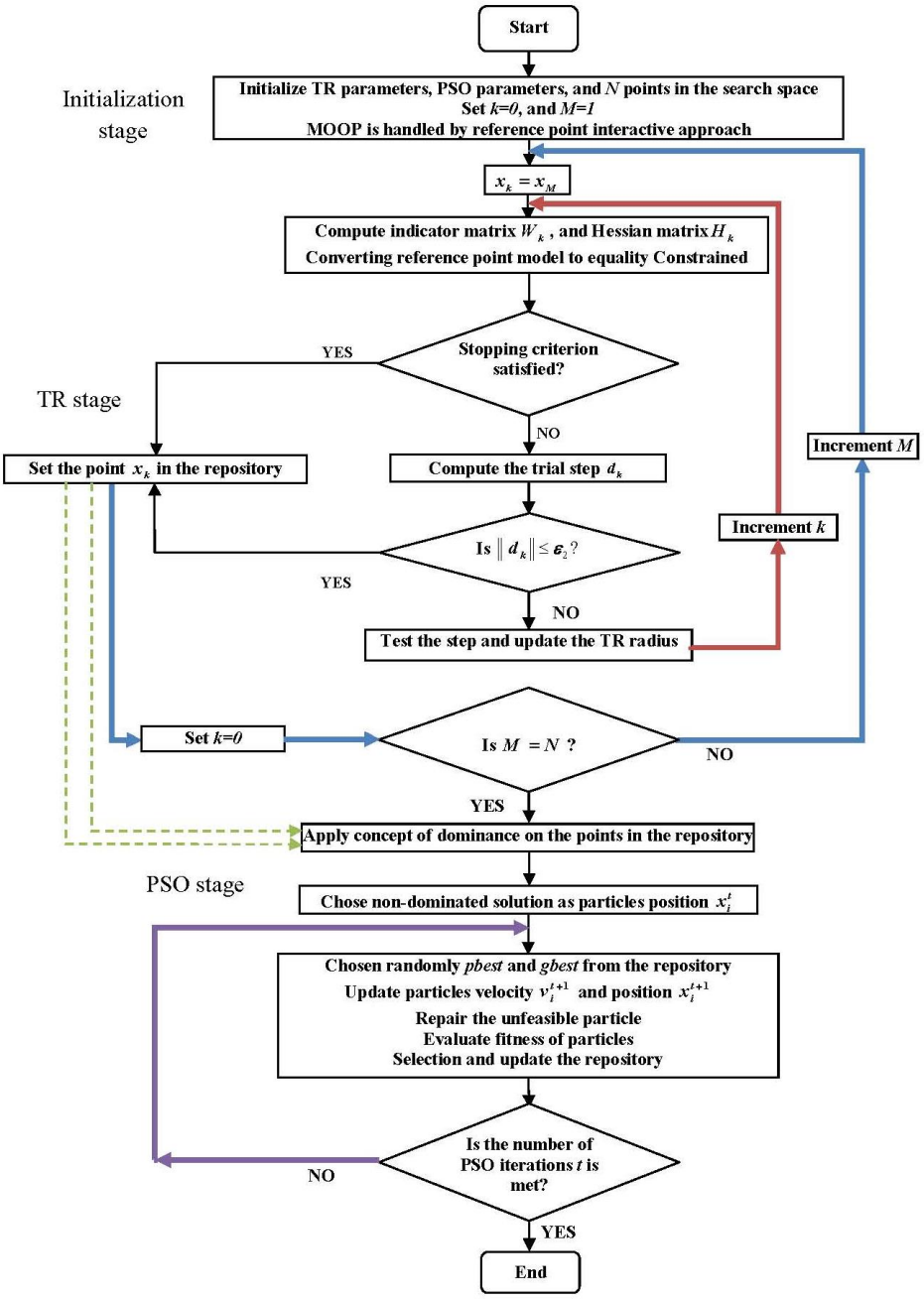

Fig. 6. Flow chart of proposed algorithm

Table 1. The parameter adopted in the implementation of the proposed algorithm

Parameter	Value	Parameter	Value
N	20-50	Δ_{max}	$10^5\Delta_0$
ε_1, ε_2	10^{-7}	Δ_{min}	10^{-3}
τ_0	0	PSO iteration	50-200
τ_1	2	w	0.6
τ_2	0.25	c_1	2.8
τ_3	0.25	c_2	1.3
Δ_0	$(1,1.5)\times\Delta_{min}$	τ	15

Test Problem-4 (Continuous Convex):

$$\text{Minimize} \quad f_1(x) = x_1^2$$
$$\text{Minimize} \quad f_2(x) = \frac{1+x_1^2}{x_2^2}$$

Subject to:

$$x_1 \in \left[\sqrt{0.1}, 1\right]$$
$$x_2 \in \left[0, \sqrt{5}\right]$$

Test Problem-5 (Continuous Convex):

$$\text{Minimize} \quad f_1(x) = x_1^2 + x_2^2$$
$$\text{Minimize} \quad f_2(x) = (2+x_1)^2 + x_2^2$$

Subject to:

$$x_1 \in [-50, 50]$$
$$x_2 \in [-50, 50]$$

Test Problem-6 (Continuous Convex):

$$\text{Minimize} \quad f_1(x) = 4x_1^2 + 4x_2^2$$
$$\text{Minimize} \quad f_2(x) = (x_1-5)^2 + (x_2-5)^2$$

Subject to:

$$C_1(x) = (x_1-5)^2 + x_2^2 \le 25 \qquad x_1 \in [0,5]$$
$$C_2(x) = (x_1-8)^2 + (x_2+3)^2 \ge 7.7 \qquad x_2 \in [0,3]$$

Test Problem-7 (Continuous Convex):

$$\text{Minimize} \quad f_1(x) = 2 + (x_1-2)^2 + (x_2-2)^2$$
$$\text{Minimize} \quad f_2(x) = 9x_1 - (x_2-1)^2$$

Subject to:

$$C_1(x) = x_1^2 + x_2^2 \le 225 \qquad x_1 \in [-20, 20]$$
$$C_2(x) = x_1 - 3x_2 + 10 \le 0 \qquad x_2 \in [-20, 20]$$

Test Problem-8 (Discrete):

$$\text{Minimize} \quad f_1(x) = x_1$$
$$\text{Minimize} \quad f_2(x) = x_2$$

Subject to:

$$C_1(x) = x_1^2 + x_2^2 - 1 - 0.1\cos\left(16\arctan\left(x_1/x_2\right)\right) \ge 0 \quad x_1 \in [0, \pi]$$
$$C_2(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \le 0.5 \qquad x_2 \in [0, \pi]$$

Test Problem-9 (Continuous Non-convex):

$$\text{Minimize} \quad f_1(x) = -\begin{bmatrix} 25(x_1 - 2)^2 + (x_2 - 2)^2 \\ +(x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2 \end{bmatrix}$$
$$\text{Minimize} \quad f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$$

Subject to:

$$C_1(x) = x_1 + x_2 - 2 \ge 0 \qquad x_1 \in [0, 10]$$
$$C_2(x) = 6 - x_1 - x_2 \ge 0 \qquad x_2 \in [0, 10]$$
$$C_3(x) = 2 - x_2 + x_1 \ge 0 \qquad x_3 \in [1, 5]$$
$$C_4(x) = 2 - x_1 + 3x_2 \ge 0 \qquad x_4 \in [0, 6]$$
$$C_5(x) = 4 - (x_3 - 3)^2 - x_6 \ge 0 \qquad x_5 \in [1, 5]$$
$$C_6(x) = (x_5 - 3)^2 + x_6 - 4 \ge 0 \qquad x_6 \in [0, 10]$$

RESULTS AND DISCUSSION

The proposed approach able to obtain the Pareto front of these kind of problems as shown **in Fig. 7 -15**.

For the test problems 1-5 (**Fig. 7-11**) we can see that our approach able to find well distribution of the Pareto-optimal curve in the objective space. Also, it is observed that the resulting Pareto front is smooth, uniformly distributed and it achieves very good solutions at the two ends of the curve.

The test problem 6 (**Fig. 12**) and the test problem 7 (**Fig. 13**) are fairly simple in that the constraints may not introduce additional difficulty in finding the Pareto-optimal solutions. It can be observed that our approach perform well and have a dense sampling of solutions along the true Pareto optimal curve.

The test problem 8 (**Fig. 14**) and the test problem 9 (**Fig. 15**) are relatively difficult. The constraints in the test problem 8 make the Pareto-optimal set discontinuous. The constraints in the test problem 9 divide the Pareto-optimal set into five regions.

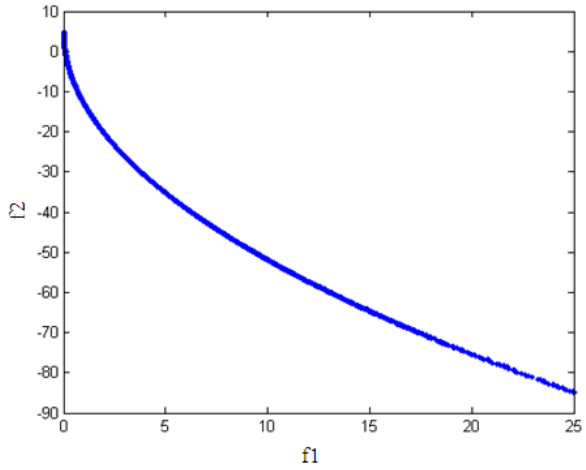

Fig. 7. Pareto front for the Test Problem (1)

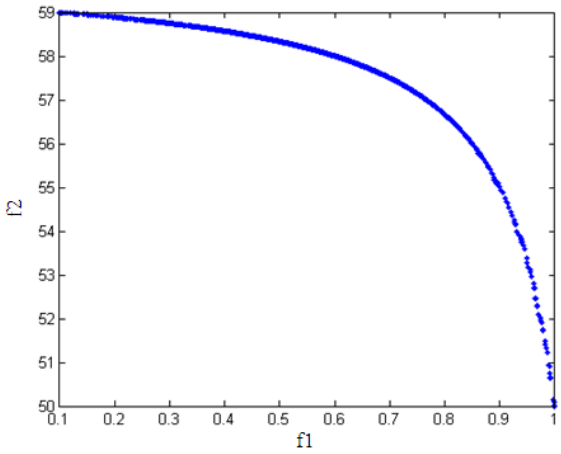

Fig. 8. Pareto front for the Test Problem (2)

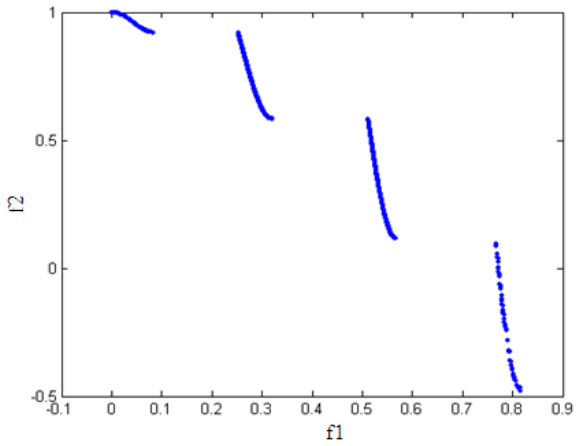

Fig. 9. Pareto front for the test problem (3)

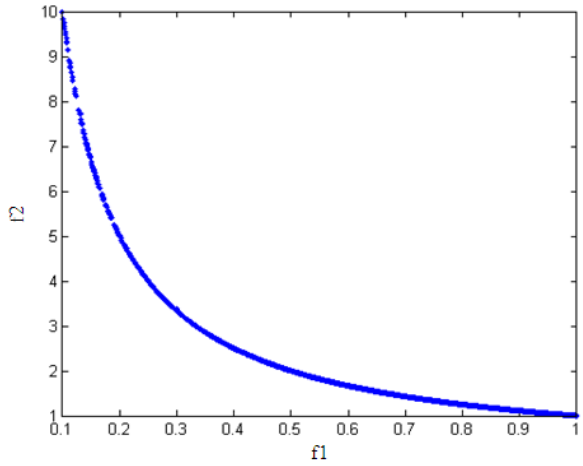

Fig. 10. Pareto front for the test problem (4)

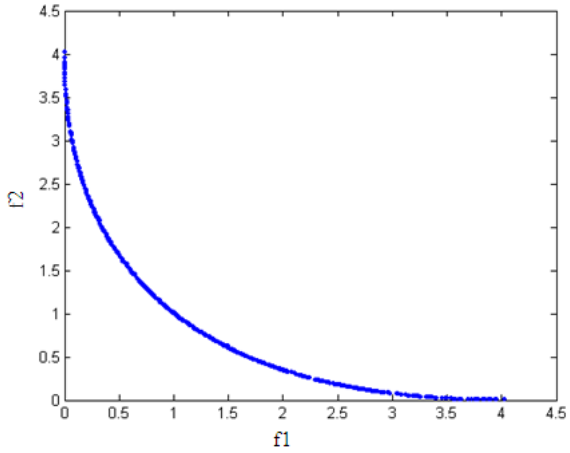

Fig. 11. Pareto front for the Test Problem (5)

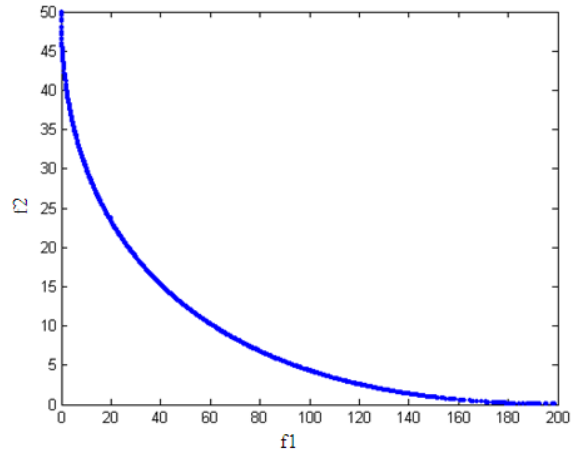

Fig. 12. Pareto front for the Test Problem (6)

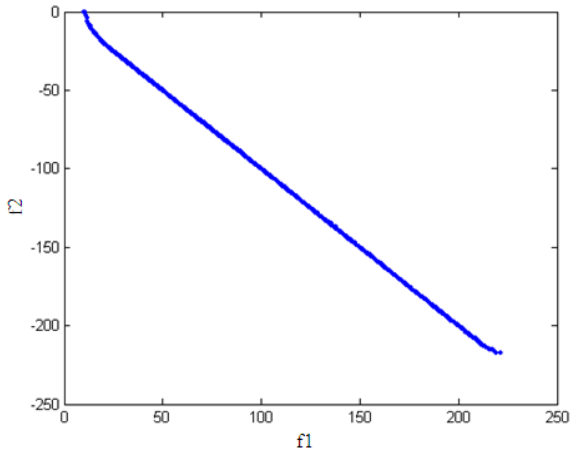

Fig. 13. Pareto front for the Test Problem (7)

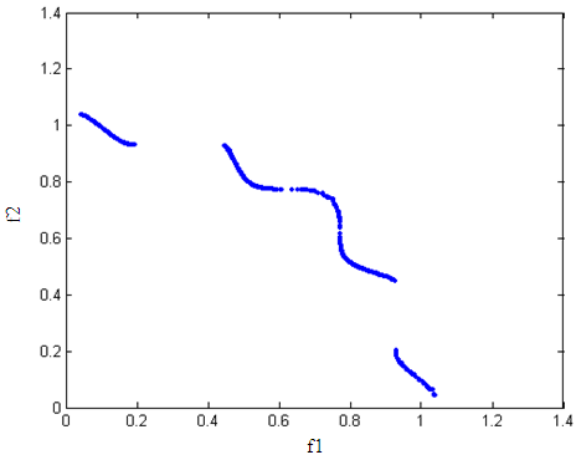

Fig. 14. Pareto front for the Test Problem (8)

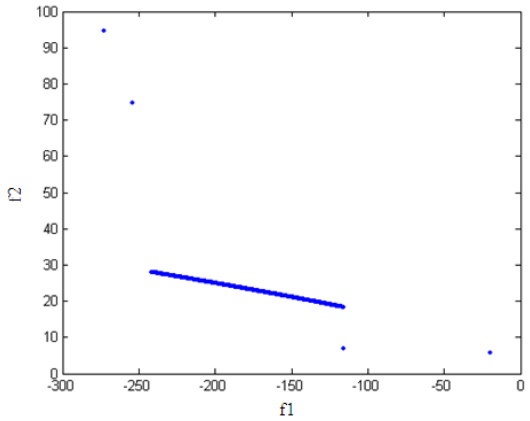

Fig. 15. Pareto front for the Test Problem (9)

As it can be seen from the graphs for the TNK problem (**Fig. 14**), our approach and displayed a better distribution of the Pareto optimal points and there are no gaps between the nondominated solutions which Making the curve is smooth. For the OSY problem (**Fig. 15**), it can be seen that our approach gave a good sampling of points at the mid-section of the curve and found a few points in the rest of the curve.

Generally we can say that the results have demonstrated that the proposed algorithm can successfully find the Pareto optimal for all test problems except test problem 9; where it's Pareto is nonconvex in the objective space. As we know that the classical techniques aim to give a single point (solution) at each run of problem solving but, the proposed approach generates the set of Pareto optimal solution, which provides the facility to save computing time.

Table 2. The GD criterion for test problems

Test problem	Generational Distance (GD)
Test problem (1)	0.00010458
Test problem (2)	0.00655497
Test problem (3)	0.00569784
Test problem (4)	0.00549784
Test problem (5)	0.00012578
Test problem (6)	0.00048679
Test problem (7)	0.00026457
Test problem (8)	0.00075481
Test problem (9)	0.01587945

Performance Assessments

There are usually two important aspects of MOO performance. One is the spread across the Pareto optimal front and the other is the ability to attain the global optimum or final tradeoffs. Every MO optimizer should have the ability of exploration and exploitation to achieve these two goal simultaneously. There are several metrics to express these two aspects with a quantitative assessment.

To evaluate the proposed algorithm, the Generational Distance (GD) criterion is used Kim and Ryu (2011). When the optimal Pareto set is known, GD is a way of estimating how far are the elements in the set of nondominated vectors found so far from those in the Pareto optimal set and is defined as follows:

$$GD = \frac{\sqrt{\sum_{i=1}^{Nv} d_i^{Nv}}}{Nv} \qquad (21)$$

where, N_v is the number of vectors in the set of nondominated solutions found so far and d_i is the Euclidean distance between each of these and the nearest member of the Pareto optimal set. If all the solution candidates are in the Pareto optimal set, then the value of GD is 0.

Table 2 shows the GD criterion for the nine test problems. In **Table 2**, we can see that GD for the first 8 problems is very small that mean the approximate Pareto obtained by the proposed approach is very near to the true Pareto solution. On the other hand, for test problem (9) we can see that GD criterion is greater than the other test problems. This is due to that this problem has nonconvex Pareto solution.

CONCLUSION

This study presents a hybrid algorithm combining TR and PSO for solving MOOPs. It is a new algorithm that performs random searching with deterministic searching and integrates the merits of both TR and PSO. In the proposed algorithm, MOOP converting to SOOP, TR is used to obtain a point on the Pareto frontier and homogeneous PSO with a dynamic constriction factor is applied to get all the points on the Pareto frontier. Various kinds of MO benchmark problems showed the effectiveness of the new algorithm and illustrate the successful result in finding a Pareto optimal set. The following are the significant contributions.

- The present work addressed an important task of combining TR with PSO to not find a single optimal solution, but to find a set of nondominated solutions
- Using the randomicity PSO and the high efficiency of TR method, can overcome the limitation of TR method and solve efficiently a class of MOOPs
- The proposed algorithm does not have any restrictions on the number of the Pareto optimal solutions found; where it keeps track of all the feasible solutions found during the optimization
- The proposed approach can be solve nonconvex MOOPs but cannot be generate all Pareto points on the frontier
- The numerical results reveal that the proposed approach can generate well-distributed sets of Pareto points very efficiently and is thus very suitable for engineering MOOPs and has good application value
- Using the GD criterion show that the proposed algorithm give good approximation of the Pareto optimal solution.

- When the initial repository has less than 3 Pareto solutions, the good result couldn't be expected and If the initial Pareto solutions saved in repository have good diversity, then this algorithm have a better results

Further research will concentrate on the possibilities to extend the proposed technique to deal more nonconvex MOOPs by using another method (i.e., hybrid method) for converting MOOP to SOOP.

REFERENCES

Y. Ou, 2011. A hybrid trust region algorithm for unconstrained optimization, Applied Numerical Mathem., 61: 900-909. DOI:10.1016/j.apnum.2011.03.002

Ahookhosh, M., K. Amini and M.R. Peyghami, 2012. A nonmonotone trust-region line search method for large-scale unconstrained optimization. Applied Mathem. Modell., 36: 478-487. DOI: 10.1016/j.apm.2011.07.021

Ahookhosh, M. and K. Amini, 2010. A Nonmonotone trust region method with adaptive radius for unconstrained optimization problems. Comput Mathem. Applications 60: 411-422. DOI: 10.1016/j.camwa.2010.04.034

Zhang, J., K. Zhang and S. Qu, 2010. A nonmonotone adaptive trust region method for unconstrained optimization based on conic model. Applied Mathem. Computation 217: 4265-4273. http://www.tandfonline.com/doi/abs/10.1080/10556780410001697677?journalCode=goms20#preview

El-Sobky, B., 2012. A multiplier active-set trust-region algorithm for solving constrained optimization problem, Applied Mathem. and Computation 219: 928–946. http://dx.doi.org/10.1016/j.amc.2012. 06.072

Kim, S. and J. Ryu, 2011. A trust-region algorithm for bi-objective stochastic optimization. Procedia Comput. Sci., 4: 1422-1430. DOI: 10.1016/j.procs.2011.04.153

Kennedy, J., R.C. Eberhart and Y. Shi, 2001. Swarm Intelligence. Morgan Kaufmann,

Parsopoulos, K.E. and M.N. Vrahatis, 2002. Particle swarm optimization method in multiobjective problems. Proceedings of the ACM 2002 Symposium on Applied Computing, pp: 603-607. DOI: 10.1145/508791.508907

Mousa, A.A., M.A. El-Shorbagy and W.F. Abd-El-Wahed, 2012. Local search based hybrid particle swarm optimization algorithm for multiobjective optimization. Swarm Evolutionary Computation 3: 1-14. DOI: 10.1016/j.swevo.2011.11.005

Figueira, J.R., A. Liefooghe, E.-G. Talbi, and A.P. Wierzbicki, 2010. A parallel multiple reference point approach for multi-objective optimization. European Journal of Operational Research 205: 390–400. http://dx.doi.org/10.1016/j.ejor.2009.12. 027

Sierra, M.R. and C.C. Coello, 2006. Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int. J. Computational Intell. Res., 2: 287-308.

Tang, L., 2013. A Hybrid Multiobjective Evolutionary Algorithm for Multiobjective Optimization Problems. IEEE Transactions on Evolutionary Computation 17: 20 – 45.

Friedrich T., T. Kroeger, and F. Neumann, 2013. Weighted preferences in evolutionary multi-objective optimization. Int. J. Mach. Learn. & Cyber. 4: 139-148. DOI: 10.1007/s13042-012-0083-y

Dennis, J., M. El-Alem and K. Williamson, 1999. A trust-region approach to nonlinear systems of equalities and inequalities. SIAM J. Optimization, 9: 291-315. DOI: 10.1137/S1052623494276208

Abd-El-Wahed, W.F., A.A. Mousa and M.A. El-Shorbagy, 2011. Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems. J. Computational Applied Mathem., 235: 1446-1453. DOI: 10.1016/j.cam.2010.08.030

Deb, K., 2001. Multi-objective Using Evolutionary Algorithms. 1 st Edn., John Wiley & Sons, LTD, New York.