Computational Intelligence

AN INTERNATIONAL JOURNAL

Computational Intelligence, Volume 0, Number 0, 2016

IMPROVING THE LOCAL SEARCH ABILITY OF SPIDER MONKEY OPTIMIZATION ALGORITHM USING QUADRATIC APPROXIMATION FOR UNCONSTRAINED OPTIMIZATION

KAVITA GUPTA,[1] KUSUM DEEP,[1] AND JAGDISH CHAND BANSAL[2]

[1]*Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand, India*
[2]*Department of Applied Mathematics, South Asian University, Akbar Bhawan, Chankyapuri, New Delhi, India*

Spider monkey optimization (SMO) algorithm, which simulates the food searching behavior of a swarm of spider monkeys, is a new addition to the class of swarm intelligent techniques for solving unconstrained optimization problems. The purpose of this article is to study the performance of SMO after incorporating quadratic approximation (QA) operator in it. The proposed version is named as QA-based spider monkey optimization (QASMO). An experimental study has been carried out to check the validity and applicability of QASMO. For validation purpose, the performance of QASMO is tested over a benchmark set of 46 scalable and nonscalable problems, and results are compared with the original SMO algorithm. In order to test the applicability of the proposed algorithm in solving real-life optimization problems, one of the most challenging optimization problems, namely, Lennard–Jones (LJ) problem is considered. LJ clusters containing atoms from three to ten have been taken into consideration, and results are presented. To the best of our knowledge, this is the first attempt to apply SMO and its proposed variant on a real-life problem. The results demonstrate that incorporation of QA in SMO has positive effects on its performance in terms of reliability, efficiency, and accuracy.

Received 5 March 2015; Revised 10 August 2015; Accepted 7 September 2015

Key words: spider monkey optimization, quadratic approximation, swarm intelligent techniques, unconstrained optimization, global optimization, Lennard–Jones problem.

NOMENCLATURE

N	swarm size
D	no. of dimensions
$U(a,b)$	uniformly generated random number between a and b
NG	number of groups in the current swarm
MG	maximum number of groups allowed in the swarm
Pr	perturbation rate
$GS[k]$	number of members in the kth group
$G[k][0]$	index of the first member of the kth group in the swarm
$G[k][1]$	index of the last member of the kth group in the swarm
SM_i	position vector of the ith spider monkey in the swarm
SM_{new}	a trial vector for creating a new position of a spider monkey
SM_{newlocal}	a trial vector for creating a new position of a spider monkey in local leader phase
$SM_{\text{newglobal}}$	a trial vector for creating a new position of a spider monkey in global leader phase
SM_r	position vector of randomly selected member of the group
$SM_{\text{worstglobal}}$	position vector of worst member of the swarm in global leader learning phase
$SM_{\text{worstlocal}}$	position vector of worst member of a group in local leader learning phase

Address correspondence to Kavita Gupta, Department of Mathematics, Indian Institute of Technology, Roorkee, Uttarakhand 247667, India; e-mail: gupta.kavita3043@gmail.com

LL_k	position vector of local leader of the kth group
GL	position vector of global leader of the swarm
$sm_{\min j}$	lower bound on the jth decision variable
$sm_{\max j}$	upper bound on the jth decision variable
sm_{ij}	jth decision variable of the ith spider moneky
fit_i	fitness of the position of the ith spider monkey
$maxfit$	best fitness value
$prob_i$	probability of the position of the ith spider monkey
$GLlt$	global leader limit
$LLlt$	local leader limit
GLC	limit count of global leader
LLC_k	limit count of local leader of the kth group

1. INTRODUCTION

Although various stochastic algorithms such as genetic algorithms (GA) (Holland 1975), controlled random search (Price and Szego 1978), particle swarm optimization (PSO) (Kennedy and Eberhart 1995), differential evolution (DE) (Storn and Price 1997), artificial bee colony (Karaboga 2005), ant colony optimization (Dorigo 1992), biogeography-based optimization (Simon 2008), harmony search algorithm (Yang 2009), bacterial foraging optimization (Passino 2002), and so on have shown good performance in solving real-world optimization problems, yet in view of no free lunch theorem by Wolpert and Macready (1997), there is no such algorithm, which can give better performance than other algorithms on all the optimization problems. That is, there is no universally accepted algorithm for all the optimization problems. Some algorithms perform better than others on some set of problems, while others perform better on some different set of problems. Thus, research on developing optimization algorithms, which can perform better on majority of the optimization problems of interest, goes on.

Spider monkey optimization (Bansal et al. 2014) is a new swarm intelligent technique inspired by the food searching strategy of spider monkeys for solving global optimization problems. Like any other swarm intelligent technique, SMO starts with the initialization of few control parameters and randomly generated solutions. The potential solutions of the swarm are represented by position of spider monkeys in the swarm. Spider monkeys update their position with each passing generation according to the updation criterion to explore and exploit different regions of the search space to obtain near-optimal solution. In SMO, every new position generated for a spider monkey by position update process is compared with the old position, and the better one is adopted. Also, SMO has been designed in such a way that it has the ability to handle two of the major problems of swarm intelligent techniques, that is, premature convergence and stagnation, in an efficient manner. Since its inception, only one modified version of the original SMO has been proposed (Kumar et al. 2014). In the first article of SMO (Bansal et al. 2014), it has been shown by numerical and statistical results that SMO is a strong competitor of some state-of-the-art algorithms (Storn and Price 1997; Karaboga 2005; Clerc 2012; Hansen and Ostermeier 1996).

The present article introduces a new variant of original SMO after implementing quadratic approximation (QA) operator in it with an objective to improve its local search ability. QA provides the minimum of the quadratic curve passing through three feasible solutions in the search space. The motivation for applying QA in SMO comes from its successful incorporation in various stochastic search techniques such as controlled random search, GA, PSO, DE, and so on in the past few years. The same operator is used with the

name QA in some papers and with quadratic interpolation (QI) in others, which is available in the literature. In the present article, it has been used with the name QA operator. QA has been implemented in different metaheuristics in different ways. In Mohan and Shanker (now Deep) (1994), QA has been used in the local phase of random search technique to replace worst feasible solution of the population. In Pant et al. (2007), QA has been used as a nonlinear crossover operator to produce an offspring from three feasible solutions say two randomly selected particles and best particle of the swarm. In Deep and Das (2008), QA has been used as an additional operator in GA. After every cycle of GA operators is completed, QA operator has been used to gain local refinements. In Deep and Bansal (2009), QA has been used with two variants of PSO, namely, PSO-W (PSO with time varying inertia weight) and PSO-C (PSO with constriction factor). In Pant et al. (2009), QA named as QI has been used to initialize the population in DE algorithm. In Liu et al. (2014), QI has been used with global version of orthogonal learning-based PSO (QIOLPSO-G).

In the present article, QA has been applied to make efficient use of the available information about the current global best and local best solutions. The neighborhood of these solutions have been searched for better solutions using QA operator. Investigation has been made on the performance of the proposed algorithm by testing it over a set of benchmark problems, and results are compared with that of the original SMO. The proposed algorithm and experimental results have been discussed in detail in the next sections.

The article is organized as follows. Section 2 presents the social behavior and food searching strategy of spider monkeys followed by description of SMO algorithm. In Section 3, QA operator and its implementation in SMO has been discussed. Section 4 provides the experimental setup followed by experimental results and discussion in Section 5. In Section 6, application of SMO and the proposed algorithm on the Lennard–Jones (LJ) cluster problem has been discussed. Finally, the article has been concluded with future scope in Section 7.

2. SPIDER MONKEY OPTIMIZATION

A detailed description of SMO has been given in the succeeding sections.

2.1. Motivation

Spider monkey optimization algorithm is inspired from the foraging and social behavior of spider monkeys. Spider monkeys are social animals which live in a group of 40–50 individuals. Their food searching strategy make them fall into the category of fission–fusion social structure-based animals. The animals of this category divide themselves into subgroups to forage at different places and then recombine to share the collected food. A female leader is responsible for the food availability of group members. In case if she does not find enough food for the group, she divides the group into subgroups. These subgroups forage in different directions during day time and night time, and they all gather at a common place to share their collected food. Based on the intelligent swarming behavior and food searching strategy exhibit by spider monkeys swarm, SMO technique has been developed.

2.2. Spider Monkey Optimization Technique

Spider monkey optimization, like all other swarm intelligent techniques, is inspired from the food searching behavior of living creatures. In SMO, food searching strategy of spider monkeys is simulated to develop this optimization technique. In addition to initialization of the swarm, SMO has six iterative steps, namely, local leader phase, global leader phase, global leader learning (GLL) phase, local leader learning (LLL) phase, local leader decision

phase, and global leader decision phase. There are four control parameters of SMO, namely, perturbation rate (*Pr*), maximum number of groups (*MG*), global leader limit (*GLlt*), and local leader limit (*LLlt*) . Their details and use in different phases has been discussed in the following paragraphs. The key point of SMO is that it has been designed in such a manner that it can handle the problem of stagnation and premature convergence, which are two of the most severe problems faced by metaheuristics, in a very efficient manner. *GLlt* and *LLlt* keep check on stagnation in the whole swarm and in the local groups, respectively, and prevents premature convergence by taking necessary steps described in GLD phase and local leader decision phase.

Before the description of each step of the algorithm in detail, here are some of the terms, which have been used during iterative steps in different phases. A D-dimensional trial vector say $SM_{\text{new}} = (sm_{\text{new1}} sm_{\text{new2}}, \ldots, sm_{\text{newD}})$ for creating a new position of a spider monkey. $LL_k = (ll_{k1} ll_{k2}, \ldots, ll_{kD})$ is the position of the local leader of the kth group and $GL = (gl_1 gl_2, \ldots gl_D)$ is the position of the global leader of the entire swarm. Also, it should be noted that updation in the position of a spider monkey is carried out dimensionwise. If during updation process, the value of a decision variable is out of predefined limits (lower and upper bounds), then the value of that decision variable can be set to the predefined limit or randomly between the predefined limits

Detailed description of each phase is given as follows:

2.2.1. Initialization of the Swarm. During initialization of the swarm, SMO generates a uniformly distributed D-dimensional initial positions vectors of N spider monkeys. The position $SM_i = (sm_{i1}, sm_{i2}, \ldots, sm_{iD})$ of ith spider monkey is initialized as follows:

$$sm_{ij} = sm_{\min j} + U(0,1) \times (sm_{\max j} - sm_{\min j}) \text{ where } i = 1, 2, \ldots, N \text{ and } j = 1, 2 \ldots, D \quad (1)$$

2.2.2. Local Leader Phase. In this phase, a new trial position say $SM_{\text{newlocal}} = (sm_{\text{newl1}} sm_{\text{newl2}}, \ldots, sm_{\text{newlD}})$ is generated for each spider monkey based on the information of its current position, local leader's experience, and local group members' experience.

Pseudocode of position update process in this phase is given in Figure 1. In Figure 1, $SM_r = (sm_{r1} sm_{r2}, \ldots sm_{rD})$ is the position of the randomly selected member of the current group. Although it is a randomly selected member of the group, yet it should be different from the member of the group that has to be updated. The amount of change in the current position depends on the perturbation rate. It can be observed from the pseudocode of the phase that the higher the perturbation rate, less change will be possible and the lower the perturbation rate, more change is possible. Thus, perturbation rate should be chosen wisely to control exploration and exploitation in different iterations. The fitness value of the newly generated position of the ith spider monkey is compared with that of its old position. If the fitness of the newly generated position is higher than that of old position, then the spider monkey updates his position with the new one, otherwise it retains its old position.

2.2.3. Global Leader Phase. In this phase, like local leader phase, a new trial position say $SM_{\text{newglobal}} = (sm_{\text{newg1}} sm_{\text{newg2}}, \ldots, sm_{\text{newgD}})$ is created for each spider monkey. But here, in place of the local leader, the experience of the global leader is used in addition to the experience of the current member and local group members' experience. In this phase, a spider monkey gets a chance to update its position based on its probability, which is directly proportional to the fitness of its current position.

The probability of the ith spider monkey in the swarm has been calculated using the following expression:

```
begin:
  for k = 1:NG do
    for i = G[k][0]:G[k][1] do
        for j = 1:D do
          if U(0, 1) ≥ Pr then
                sm_newj = sm_ij + U(0,1) × (ll_kj − sm_ij) + U(−1,1) × (sm_rj − sm_ij)
                 else
                  sm_newj = sm_ij
            end if
          end for
          if (fit(SM_new)>fit(SM_i))
             SM_i = SM_new
          end if
      end for
   end for
end
```

FIGURE 1. Pseudocode for *local leader phase*.

```
begin:
  for k = 1:NG do
     count = 1;
     while count < GS[k] do
        for i = G[ k][0]:G[k][1] do
          if (U(0, 1) < prob[i]) then
                count = count + 1.
                Randomly select j ∈ {1...D}.
                Randomly select SM_r from kth group s.t. r != i.
                sm_newj = sm_ij + U(0,1) × (gl_j − sm_ij) + U(−1,1) × (sm_rj − sm_ij)
            end if
            if(fit(SM_new)>fit(SM_i))
               SM_i = SM_new
            end if
        end for
        if (i= =G[k][1]) then
            i = G[k][0]
        end if
      end while
  end for
end
```

FIGURE 2. Pseudocode for *global leader phase*.

$$prob_i = 0.9 * \left(\frac{fit_i}{maxfit}\right) + 0.1 \tag{2}$$

Pseudocode for position update process of global leader phase is given in Figure 2.

From the update process in this phase, it is clear that spider monkeys having higher fitness values possess better chance to improve their position as compared with other members of the swarm

2.2.4. Global Leader Learning Phase. In this phase, position of the global leader is updated by applying greedy selection in the swarm. A spider monkey whose position is having best fitness value will be updated as global leader of the swarm. This phase has

```
begin :
   //update position of the global leader of the swarm by applying greedy selection
    if(position of global leader is updated from previous position) then
        GLC = 0
    else
        GLC = GLC + 1
   end if
end
```

FIGURE 3. Pseudocode for *global leader learning phase*.

```
begin:
   for k=1:NG do
     //update position of the leader of the group
      if(position of local leader is updated from previous position) then
         LLC_k = 0
      else
         LLC_k = LLC_k + 1
      end if
    end for
 end
```

FIGURE 4. Pseudocode for *local leader learning phase*.

```
begin:
   for k = 1:NG do
        if (LLC_k > LLlt) then
                LLC_k = 0
                   for i=G[k][0]:G[k][1] do
                        for j=1:D do
                             if (U(0, 1) ≥ Pr) then
                                  sm_ij = sm_minj + U(0,1) × (sm_maxj − sm_minj)
                              else
                                 sm_ij = sm_ij + U(0,1) × (gl_j − sm_ij) + U(0,1) × (sm_ij − ll_kj)
                             end if
                       end for
                   end for
          end if
    end for
 end
```

FIGURE 5. Pseudocode for *local leader decision phase*.

been described in Figure 3. After the updation of the global leader, it is checked whether the position of the global leader has been changed or not. *GLC* records how many times position of the global leader has not been updated since the last updation.

2.2.5. Local Leader Learning Phase. In this phase, the position of every local leader is updated by applying greedy selection in that group. A spider monkey whose position is having the best fitness value in the group will be updated as local leader of the group.

```
begin:
    if (GLC > GLlt) then
       GLC = 0
        if (NG < MG) then
             NG=NG+1
        else
             NG=1
        end if
        Apply Local leader learning phase
    end if
end
```

FIGURE 6. Pseudocode for *global leader decision phase*.

```
begin:
  Initialize the swarm using equation (1)
  Initialize LLlt, GLlt, Pr, MG
  Iteration=0
  Calculate fitness value of the position of each spider monkey in the swarm
  Select Global Leader and Local Leaders by applying Greedy Selection
  while (termination criterion is not satisfied) do
      //Local Leader Phase
     //Calculate Probabilities
    //Global Leader Phase
    //Global Leader Learning Phase
    //Local Leader Learning Phase
    //Local Leader Decision Phase
    //Global Leader Decision Phase
      Iteration = iteration +1
  end while
end
```

FIGURE 7. Pseudocode for spider monkey optimization.

Description about this phase is shown in Figure 4. LLC_k records how many times position of the local leader of the kth group has not been updated since the last updation.

2.2.6. Local Leader Decision Phase. Local leader limit is an important control parameter associated with every local leader and helps them to take important decision regarding the group. It is predefined and checks stagnation in the group. If the position of the local leader of a particular group is not updated within the LLlt, then all the members of the group will be reinitialized. Reinitialization process is given in Figure 5.

2.2.7. Global Leader Decision Phase. Global leader limit is also a control parameter of SMO. It is associated with global leader of the swarm and helps to check stagnation in the whole swarm. Like LLlt, it is also predefined. If the position of the global leader is not updated within a GLlt, then the swarm is divided into groups to start the search in different directions. But there is a limit on the number of groups in which the whole swarm can be divided. This limit is defined by the control parameter named as maximum number of groups. It is also predefined. If there are already maximum number of groups in the swarm, then all the groups get merged into a single group and the division of groups repeats. This procedure has been given in Figure 6.

Pseudocode of SMO is given in Figure 7.

3. QUADRATIC APPROXIMATION-BASED SMO

3.1. Working of QA

Quadratic approximation has been incorporated in the basic version of SMO with the objective to improve its local search ability. QA provides the point of minima of the quadratic curve passing through three solutions. QA works in the following manner.

First, three feasible solutions say, $A(a_1a_2,\ldots,a_D)$ with the best fitness value, $B(b_1b_2,\ldots,b_D)$ and $C(c_1c_2,\ldots,c_D)$ are randomly chosen such that A, B, and C all are distinct. Then a new solution P$(p_1p_2,\ldots,p_D)$, which is the point of minima of the quadratic curve passing through A, B, and C is given by

$$p[j] = \frac{1}{2}\frac{\left(b_j^2 - c_j^2\right) f\,(A) + \left(c_j^2 - a_j^2\right) f\,(B) + \left(a_j^2 - b_j^2\right) f(C)}{\left(b_j - c_j\right) f\,(A) + \left(c_j - a_j\right) f\,(B) + \left(a_j - b_j\right) f(C)} \quad \forall j = 1, 2, \ldots, D \tag{3}$$

where $f(A)$, $f(B)$, and $f(C)$ are the values of objective function f at A, B, and C, respectively.

3.1.1. Implementation Strategy of QA in SMO. QA has been implemented in the GLL phase and the LLL phase. In GLL phase, the three solutions will be two randomly selected members from the swarm in addition to the global leader. In LLL phase, these three solutions will include the local leader of the group and two randomly selected members from the same group. The reason for selecting these two phases for incorporating QA is that in these two phases, we obtain updated position of global and local leaders and the probability of obtaining better solutions in the neighborhood of good solutions is higher as compared with other solutions. Modified GLL phase and modified LLL phase has been described in detail in the next paragraphs.

In *modified GLL phase*, first, the position of the global leader of the swarm is updated. Now, the position say $SM_{\text{worstglobal}}$ of the worst member (solution having the minimum fitness value) of the swarm is found. Then, solutions are generated using QA for a predefined number of times till we obtain a solution, which is better than the worst member of the swarm. For this, we choose three points A, B, and C from the swarm, where A = GL, and B and C are positions of randomly chosen members of the swarm such that A, B, and C are all distinct. Generate a new position using equation (2). If the fitness value of a newly generated position is better than that of the worst member of the group, then update the worst position with a new one. Pseudocode for this modified phase is given in Figure 8.

In *modified local leader learning phase*, the process described in the aforementioned paragraph repeats itself for every group. Here, the whole swarm is replaced by a particular group, and the global leader is replaced by a local leader. Figure 9 provides pseudocode of position update process in this phase.

4. EXPERIMENTAL SETUP

4.1. Benchmark Set

In the field of nature-inspired algorithms, it is a common practice to test and compare the performance of different algorithms over a benchmark set. An effort to define a benchmark set such that a particular algorithm performs better than other algorithms in all the problems in the benchmark set under consideration is of less importance. Rather, a

```
begin:
   Update the position of the global leader in the swarm
   find SM_worstglobal
     for i = 1:1000 do
            //Select A= GL, B and C are positions of randomly chosen members of the
            swarm   such that A, B, C all are distinct
          //generate P using equation (3)
          if (fit(P)>fit(SM_worstglobal))
            SM_worstglobal = P
            Terminate the loop
          end if
      end for
      if(fit(P)>fit(GL))
        GL=P
      end if
      If(position of global leader is updated from previous position) then
        GLC = 0
      else
        GLC = GLC + 1
      end if
end
```

FIGURE 8. Pseudocode for the *modified global leader learning phase*.

```
begin:
  for k=1:NG do
    Update the position of the local leader in the swarm
    find SM_worstlocal
     for i = 1:1000 do
            //Select A = LL_k , B and C are positions of randomly chosen members of the
                                                    groups such that A,B,C  all are distinct
          //generate P using equation (3)
          if(fit(P)>fit(SM_worstlocal))
            SM_worstlocal= P
            Terminate the loop
          end if
      end for
      if (fit(P)>fit(LL_k))
        LL_k=P
      end if
      if (position of local leader is updated from previous position) then
        LLC_k = 0
      else
        LLC_k = LLC_k+ 1
      end if
  end for
end
```

FIGURE 9. Pseudocode for the *modified local leader learning phase*.

benchmark set should be chosen in such a way to include problems of mixed characteristics so that it becomes easier to make a conclusion about the type of problems in which a particular algorithm performs better than the other algorithms. Problems with varying characteristics helps in exploring the strengths and weaknesses of an algorithm. For example, unimodal problems are considered to check the convergence speed of an algorithm. Multimodal problems are included to test algorithm's ability to escape the trap of local minimas. Problems with flat surfaces are considered to judge algorithm's search ability in the absence

of guiding search directions. In order to check the performance of the proposed algorithm QASMO, experiments have been performed over a large set of 46 benchmark problems. In this benchmark set, in addition to all the 26 benchmark problems from the paper of original SMO (Bansal et al. 2014), some additional benchmark problems from the literature have also been considered for an extensive analysis of the performance of the proposed algorithm. This benchmark set is large enough to include problems having objective functions of different characteristics such as unimodal, multimodal, separable, nonseparable, discontinuous, and so on. This benchmark set is broadly divided into two categories, namely, scalable and nonscalable problems. Out of these 46 problems, the first 30 problems are scalable, and the rest of the problems are nonscalable. All the problems are of minimization type. List of problems along with their search range and objective function is provided in the Appendix. For scalable problems, lower and upper bounds on the decision variables are the same for each dimension of the solution. Also, lower and upper bounds of each decision variable are the same in nonscalable problems except in the two cases: problem nos. 32 and 36.

4.2. Implementation, Parameter Setting, and Termination Criterion

Spider monkey optimization and QASMO have been implemented in C. A total of 100 independent runs have been performed for each SMO and QASMO. Every run starts with a different initial swarm. To make sure that both the algorithms obtain the same initial swarm in every run, the same seed has been used by both the algorithms to start the same run.

Parameter setting for both the algorithms is the same to make a fair comparison in their performance. Performance of an algorithm largely depends on fine tuning of parameters. Fine-tuning of parameters means finding the most appropriate values of the control parameters of an algorithm. The difficulty of selecting these parameters increases with the increase in the number of problems in the benchmark set because of the presence of different types of objective functions. Thus, finding a common set of parameters is a challenging and necessary task in an algorithm.

Out of the four control parameters (pr, MG, LLlt, and GLlt), parameter setting for the three control parameters (pr, MG, and GLlt) has been adopted from the original article of SMO (Bansal et al. 2014). Local leader limit has been set to 100 instead of 1500 to encourage more exploration. Also, the swarm size has been taken 150 in this article in place of the swarm size, which is 50 in Bansal et al. (2014) to have a better exploration of the search space. In the original article of SMO, acceptable error for each benchmark problem is different, and the reason of choosing them differently is not specified. Thus, to maintain some uniformity in the termination criterion, the same acceptable error has been defined for each benchmark problem in the benchmark set. Following parameter setting and termination criterion is adopted for experiment:

Swarm size (fixed) =150
Perturbation rate (pr) (changing with iterations) = linearly increasing ([0.1, 0.4])
Maximum number of groups (MG) (fixed) =5
Local leader limit (fixed) = 100
Global leader limit (fixed) = 50
Total number of runs =100
Maximum number of iteration = 4000
acceptable error =1.0e-05
Stopping criterion = either maximum number of iterations are executed or acceptable error is achieved (whichever is obtained earlier)

5. EXPERIMENTAL RESULTS AND DISCUSSION

Performance of QASMO is investigated over a benchmark set of 46 problems including both scalable and nonscalable problems, and results are compared against original SMO. Comparisons have been made on the basis of reliability, efficiency, and accuracy. Reliability is measured in terms of number of successful runs out of fixed number of independent runs. A run is considered to be successful if error value of the global leader is less than or equal to the acceptable error, which is set to be 1.0e-5 here. Error value is the absolute difference between the global best value obtained and the optimal value of the problem. If a run is successful for a particular problem, the problem is said to be solved in that particular run. Efficiency is measured in terms of average number of function evaluations for successful runs, and accuracy is the degree of precision in locating the global optima, and it is measured in terms of error values obtained in all the runs. For this purpose, the number of successful runs out of the total number of independent runs, the average number of function evaluations of successful runs, and the best, average, and worst of the error values obtained in 100 independent runs have been recorded. Results of scalable and nonscalable problems have been discussed separately followed by overall conclusion of the performance of both the algorithms on both scalable and nonscalable problems. In order to test the scalability of the proposed algorithm, results of scalable problems have been recorded for 30 dimensions and 50 dimensions.

Results of both the algorithms have been analyzed in two ways. First, the impact of QA operator in SMO has been studied individually in terms of the number of successful runs, the average number of function evaluations for successful runs, and the best, average and worst of error values obtained in 100 runs. For this purpose, two comparison criteria have been defined.

Criterion 1 is used to compare the reliability and efficiency of both the algorithms. Comparison according to this criterion has been made on the basis of the following information:

- Number of successful runs out of 100 runs; and
- Average number of function evaluations of successful runs.

Solving a problem is important, so the first or higher preference is given to number of successful runs. The criterion is only applied when at least one of the algorithms is said to have positive number of successful runs.

There can be three cases:

Case1: When both the algorithms have different number of successful runs.
In this case, algorithm with more number of successful runs is the winner and is said to perform strictly better than the other.

Case 2: When both algorithms have the same number of successful runs and different number of function evaluations for successful runs.
In this case, algorithm with less number of average function evaluations for successful runs is said to perform better than the other.
Also, to further investigate the significance in the difference of average number of function evaluations of successful runs in this case, *t*-test at a significance level of 0.05 has been applied ; the symbol "=" indicates there is no significant difference between the average of function evaluations of two algorithms and the symbols "+" and "−" indicate that QASMO performs significantly better and worse than SMO, respectively.

Case 3: In extreme case, where both the algorithms have the same number of successful runs as well as the same number of average function evaluations of successful runs, both the algorithms are considered equivalent.

Criterion 2 is employed to compare the accuracy of both the algorithms. Comparison has been done on the basis of following:

- Best, average, and worst of the error values obtained in 100 runs

Following cases are possible for comparison:

TABLE 1. Number of successful runs out of 100 independent runs and average number of function evaluations for successful runs for SMO and QASMO (scalable problems – 30 dimensions).

	Number of successful runs (out of 100 runs)		Average number of function evaluations of successful runs	
Problem number	SMO	QASMO	SMO	QASMO
1	100	100	33,192	**30,626**
2	100	100	26,566	**23,254**
3	100	100	**103,625**	407,833
4	0	0	∼	∼
5	100	100	**229,495**	1,176,179
6	100	100	63,917	**59,159**
7	100	100	189,827	**85,764**
8	100	100	**24,027**	56,411
9	100	100	37,701	**33,967**
10	100	100	25,421	**23,371**
11	47	**100**	650,570	**218,639**
12	100	100	8,502	**53,700**
13	100	100	32,220	**29,952**
14	100	100	62,664	**58,450**
15	0	0	∼	∼
16	100	100	38,287	**35,335**
17	0	0	∼	∼
18	100	100	14,002	**11,860**
19	100	100	22,628	**20,988**
20	0	0	∼	∼
21	7	**42**	**1,051,267**	5,794,852
22	83	**100**	**354,061**	764,419
23	100	100	48,381	**44,862**
24	100	100	30,839	**27,865**
25	100	100	**30,695**	30,961
26	100	100	39,151	**36,457**
27	100	100	44,665	**41,482**
28	0	0	∼	∼
29	100	100	**106,373**	563,615
30	100	100	63,980	**59,646**

Better entries are shown in bold.

Case 1 : When an algorithm, say A, has all the three (best, average, and worst) values less than the other algorithm, say B, then algorithm A is said to be strictly better than algorithm B.

Case 2 : When best and average error values of A are strictly less than those of B, and the worst error value of A is equal to that of B, then A is said to perform better then B.

Case 3: When both A and B have the same best, average, and worst error values, then both A and B are said to be equivalent.

The second method of analysis is to compare the relative performance of two algorithms on the basis of performance indices (PIs) by giving weighted importance to the number of successful runs, the average number of function evaluations of successful runs and the average of error values.

The formula for calculating the values of PIs for both the algorithms have been given as

$$PI = \frac{1}{N_P} \sum_{i=1}^{N_P} \left(k_1 \alpha_1^i + k_2 \alpha_2^i + k_3 \alpha_3^i\right)$$

where $\alpha_1^i = \frac{Sr^i}{Tr^i}$,

$$\alpha_2^i = \begin{cases} \frac{Mf^i}{Af^i} & \text{if } Sr^i > 0 \\ 0 & \text{if } Sr^i = 0 \end{cases}$$

$$\alpha_3^i = \begin{cases} \frac{Me^i}{Ae^i} & \text{if } err^i > 0 \\ 1 & \text{if } err^i = 0 \end{cases}$$

i $= 1, 2, \ldots, N_P$

Sr^i number of successful runs of the ith problem

Tr^i total number of runs of the ith problem

Mf^i minimum of average number of function evaluations of successful runs used by both the algorithms for obtaining the solution of the ith problem

Af^i average number of function evaluations of successful runs used by

TABLE 2. t-test results for problems having identical number of successful runs for SMO and QASMO (scalable problems – 30 dimensions).

Problem number	Sign	Problem number	Sign
1	+	14	+
2	+	16	+
3	−	18	+
5	−	19	+
6	+	23	+
7	+	24	+
8	−	25	=
9	+	26	+
10	+	27	+
12	+	29	−
13	+	30	+

	an algorithm for obtaining solution of the ith problem
Me^i	minimum of the average of error values of the ith problem obtained by both the algorithms in total number of runs
Ae^i	average of error values of the ith problem obtained by an algorithms in total number of runs
err^i	error value of the ith problem
N_P	total number of problems evaluated

k_1, k_2, and k_3 are the weights corresponding to total number of successful runs, the average number of function evaluations of successful runs, and the average of error values such that

TABLE 3. Best, average, and worst of error values obtained in 100 independent runs (scalable problems –30 dimensions).

	Best		Average		Worst	
Problem number	SMO	QASMO	SMO	QASMO	SMO	QASMO
1	**6.01E-06**	6.56E-06	8.72E-06	**8.72E-06**	9.99E-06	**9.98E-06**
2	**3.12E-06**	4.91E-06	8.03E-06	**7.85E-06**	9.99E-06	**9.83E-06**
3	3.70E-06	**2.06E-18**	8.52E-06	**8.52E-06**	9.99E-06	**9.99E-06**
4	1.57E-02	**1.74E-03**	1.63E+01	**7.90E+00**	8.08E+01	**2.08E+01**
5	2.17E-06	**1.14E-13**	8.42E-06	**7.36E-06**	9.97E-06	**9.97E-06**
6	**7.58E-06**	7.92E-06	9.43E-06	**9.28E-06**	9.98E-06	9.98E-06
7	7.68E-06	**7.32E-06**	9.36E-06	**9.26E-06**	**9.99E-06**	1.00E-05
8	**7.54E-09**	1.46E-07	5.11E-06	**4.99E-06**	9.98E-06	**9.97E-06**
9	6.42E-06	**4.44E-16**	8.92E-06	**8.41E-06**	**9.99E-06**	1.00E-05
10	**5.15E-06**	6.30E-06	8.85E-06	**8.70E-06**	1.00E-05	**1.00E-05**
11	7.36E-06	**7.24E-06**	4.17E+00	**9.32E-06**	4.62E+01	**1.00E-05**
12	**5.62E-06**	5.97E-06	**8.74E-06**	8.81E-06	**9.98E-06**	9.99E-06
13	5.87E-06	**5.76E-06**	**8.71E-06**	8.81E-06	9.99E-06	**9.94E-06**
14	**7.44E-06**	7.52E-06	**9.30E-06**	9.33E-06	9.99E-06	**9.99E-06**
15	2.00E-01	**9.99E-02**	5.85E-01	**4.84E-01**	1.05E+01	**4.11E+00**
16	5.36E-06	**5.01E-06**	**8.63E-06**	8.77E-06	9.99E-06	**9.98E-06**
17	**9.38E-03**	5.03E-01	**1.49E+00**	2.10E+00	8.67E+00	**7.75E+00**
18	2.74E-06	**1.34E-06**	7.60E-06	**6.77E-06**	**9.97E-06**	9.98E-06
19	0.00E+00	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00	0.00E+00
20	7.73E+00	**6.86E+00**	9.73E+00	**8.59E+00**	**1.22E+01**	1.43E+01
21	7.78E-06	**0.00E+00**	**1.34E+00**	2.10E+00	**4.73E+00**	1.61E+01
22	7.58E-08	**3.63E-08**	1.54E+02	**4.89E-06**	1.45E+04	**9.89E-06**
23	**6.17E-06**	6.21E-06	8.60E-06	**8.44E-06**	**9.93E-06**	9.98E-06
24	6.17E-06	**4.38E-06**	8.77E-06	**8.74E-06**	9.99E-06	9.99E-06
25	5.98E-06	**7.29E-08**	8.78E-06	**8.50E-06**	1.00E-05	**9.98E-06**
26	5.97E-06	**4.77E-06**	8.80E-06	**8.68E-06**	9.99E-06	**9.99E-06**
27	5.67E-06	**5.47E-06**	8.79E-06	**8.51E-06**	1.00E-05	**9.99E-06**
28	1.24E+05	**1.20E+05**	1.91E+05	**1.86E+05**	2.41E+05	**2.40E+05**
29	**3.94E-06**	4.80E-06	8.67E-06	**8.66E-06**	**9.97E-06**	9.99E-06
30	**7.83E-06**	7.89E-06	9.42E-06	**9.33E-06**	**9.99E-06**	1.00E-05

Better entries are shown in bold.

$0 \leq k_1, k_2, k_3 \leq 1$ and $k_1 + k_2 + k_3 = 1$. As given in Deep and Thakur (2007), two equal weights are assigned to two variables keeping the third variable to vary from 0 to 1.

Possible cases are given by

(i) $k_1 = w, k_2 = k_3 = \frac{1-w}{2}$, $0 \leq w \leq 1$
(ii) $k_2 = w, k_1 = k_3 = \frac{1-w}{2}$, $0 \leq w \leq 1$
(iii) $k_3 = w, k_1 = k_2 = \frac{1-w}{2}$, $0 \leq w \leq 1$

In case (i), the average number of function evaluations of successful runs and the average of error values have been given equal weights. In case (ii), the number of successful runs

TABLE 4. Number of successful runs out of 100 independent runs and average number of function evaluations for successful runs for SMO and QASMO (scalable problems – 50 dimensions).

	Number of successful runs (out of 100 runs)		Average number of function evaluations of successful runs	
Problem number	SMO	QASMO	SMO	QASMO
1	100	100	56,858	**51,289**
2	100	100	50,558	**43,152**
3	100	100	**113,365**	210,051
4	0	0	∼	∼
5	**100**	96	**451,232**	2,775,798
6	100	100	106,761	**96,231**
7	95	**100**	543,832	**192,245**
8	100	100	**14,352**	22,720
9	100	100	**88,981**	93,920
10	100	100	43,824	**39,398**
11	0	**100**	∼	**555,888**
12	100	100	99,214	**9,404**
13	100	100	55,784	**50,768**
14	100	100	106,692	**96,945**
15	0	0	∼	∼
16	100	100	67,096	**60,459**
17	0	0	∼	∼
18	100	100	25,427	**18,034**
19	100	100	41,238	**38,972**
20	0	0	∼	∼
21	0	0	∼	∼
22	80	**100**	**339,593**	814,104
23	100	100	83,926	**75,603**
24	100	100	54,562	**49,861**
25	100	100	**53,508**	61,011
26	100	100	70,121	**63,906**
27	100	100	75,557	**68,645**
28	0	0	∼	∼
29	100	100	**101,583**	328,982
30	100	100	106,871	**96,701**

Better entries are shown in bold.

TABLE 5. *t*-test results for problems having identical number of successful runs for SMO and QASMO (scalable problems – 50 dimensions).

Problem number	sign	Problem number	sign
1	+	16	+
2	+	18	+
3	−	19	=
6	+	23	+
7	+	24	=
9	=	25	=
10	+	26	+
12	+	27	+
13	+	29	−
14	+	30	+

and the average error values have been given equal weights, and in case (iii), the number of successful runs and the average number of function evaluations have been given equal weights. In the figures of PI, horizontal axis displays the weight varying from 0 to 1, and vertical axis displays the PI varying from 0 to 1.

Here is the brief description of how the results have been organized in tables and in figures for comparison for both scalable and nonscalable problems.

Tables 1–3 present the numerical and statistical results for scalable (1–30) problems for 30 dimensions , Tables 4–6 present the numerical and statistical results for scalable (1–30) problems for 50 dimensions, and Tables 7–9 present the numerical and statistical results for nonscalable (31–46) problems. Better entries appear in bold. Figures 10–12 show PI graphs of scalable problems for 30 dimensions, Figures 13–15 show PI graphs of scalable problems for 50 dimensions, and Figures 16–18 show PI graphs of nonscalable problems.

There are 30 scalable problems in the benchmark set. Discussion of results of scalable problems of 30 dimensions is given in the succeeding discussions.

From Table 1, it is observed that out of 30 problems, SMO has 100 successful runs in 22 problems, while QASMO has 100 successful runs in 24 problems. In problem nos. 11, 21, and 22, QASMO performs strictly better than SMO in criterion 1. Out of the 22 problems where both the algorithms have identical number of successful runs (100 successful runs), QASMO performs better than SMO in 17 problems according to criterion 1. *t*-test result for average number of function evaluations of successful runs of both the algorithms for the problems having identical number of successful runs has been given in Table 2. Also, it can be observed in Table 1 that QASMO has more number of successful runs than SMO in problem nos. 21 and 22 although it comes at the cost of more number of function evaluations. Five problems (nos. 4, 15, 17, 20, and 28) are not solved by any of the two algorithms.

Table 2 provides *t*-test results for the average number of function evaluations of successful runs for the problems where both the algorithms have identical number of successful runs. *t*-test results convey that there is a significant difference in the average number of function evaluations in 21 problems out of 22 problems. QASMO performs significantly better than SMO in terms of convergence speed in 17 problems.

To check the solution quality obtained by both the algorithms, best, average, and worst of the error values have been provided in Table 3. For best error values, QASMO is better than SMO over 18 problems, and in problem no. 19, both the algorithms have the same error value. For average error values, QASMO is better than SMO over 21 problems, and there are three problems where both QASMO and SMO have the same average error. For worst error

TABLE 6. Best, average, and worst of error values obtained in 100 independent runs (scalable problems – 50 dimensions).

	Best		Average		Worst	
Problem number	SMO	QASMO	SMO	QASMO	SMO	QASMO
1	**6.41E-06**	7.15E-06	**9.17E-06**	9.20E-06	**9.99E-06**	1.00E-05
2	5.73E-06	**3.18E-06**	**8.63E-06**	8.74E-06	**9.96E-06**	9.98E-06
3	**6.26E-06**	7.62E-06	**9.09E-06**	9.17E-06	1.00E-05	**9.99E-06**
4	1.25E+00	**2.26E-03**	4.72E+01	**4.38E+01**	1.57E+02	**1.41E+02**
5	3.02E-06	**1.30E-11**	**8.45E-06**	2.26E+00	**1.00E-05**	1.70E+02
6	8.14E-06	**7.96E-06**	9.60E-06	**9.55E-06**	1.00E-05	1.00E-05
7	8.32E-06	**7.88E-06**	1.37E-03	**9.49E-06**	7.20E-02	**9.99E-06**
8	**1.62E-08**	6.14E-08	5.19E-06	**5.03E-06**	9.99E-06	**9.96E-06**
9	1.78E-06	**1.78E-15**	8.97E-06	**6.61E-06**	1.00E-05	**9.99E-06**
10	7.41E-06	**7.28E-06**	9.25E-06	**9.14E-06**	**9.99E-06**	1.00E-05
11	7.52E-05	**8.52E-06**	3.51E-01	**9.73E-06**	3.15E+01	**1.00E-05**
12	6.62E-06	**6.41E-06**	9.19E-06	**9.04E-06**	9.98E-06	9.98E-06
13	6.98E-06	**5.69E-06**	**9.15E-06**	9.18E-06	9.99E-06	**9.98E-06**
14	8.50E-06	**8.22E-06**	9.55E-06	**9.52E-06**	1.00E-05	1.00E-05
15	3.00E-01	**2.00E-01**	**1.01E+00**	1.23E+00	**1.41E+01**	1.44E+01
16	7.15E-06	**6.10E-06**	9.10E-06	**9.05E-06**	**9.98E-06**	9.99E-06
17	**1.65E+00**	3.61E+00	**5.37E+00**	6.25E+00	**9.53E+00**	1.08E+01
18	**1.76E-06**	2.53E-06	7.54E-06	**7.08E-06**	**9.95E-06**	9.96E-06
19	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00	0.00E+00
20	1.75E+01	**1.37E+01**	1.98E+01	**1.67E+01**	6.90E+01	**4.26E+01**
21	**1.88E+00**	2.39E+00	**5.58E+00**	5.81E+00	2.32E+01	**2.17E+01**
22	1.16E-07	**7.75E-08**	6.48E+02	**4.55E-06**	6.30E+04	**9.94E-06**
23	6.85E-06	**6.46E-06**	9.21E-06	**8.95E-06**	**9.99E-06**	1.00E-05
24	6.59E-06	**3.18E-06**	9.10E-06	**9.06E-06**	1.00E-05	**9.99E-06**
25	6.32E-06	**5.42E-08**	9.09E-06	**8.51E-06**	**9.99E-06**	1.00E-05
26	7.14E-06	**6.44E-06**	9.11E-06	**8.99E-06**	**9.94E-06**	1.00E-05
27	7.08E-06	**5.98E-06**	**9.07E-06**	9.08E-06	9.99E-06	**9.98E-06**
28	9.15E+07	**8.17E+07**	**1.22E+08**	1.23E+08	1.56E+08	**1.54E+08**
29	**2.61E-06**	5.95E-06	9.07E-06	**8.92E-06**	**9.98E-06**	1.00E-05
30	**8.20E-06**	8.40E-06	9.51E-06	**9.51E-06**	9.99E-06	9.99E-06

Better entries are shown in bold.

values, QASMO is better than SMO over 13 problems, and both algorithms have the same worst error value on eight problems. There are 12 problems in total where the performance of QASMO is better than or equal to in all the three cases.

In Table 3, it can be seen that QASMO is strictly better than SMO over eight problems (nos. 4, 11, 15, 22, and 25–28) according to criterion 2. Both the algorithms perform identically on problem 19. There are three problems (nos. 3, 5, and 24) where QASMO performs better than SMO. In problem no. 12, SMO performs strictly better than QASMO, and in problem no. 14, SMO performs better than QASMO.

In Tables 1 and 3, it can be concluded that in five problems where both the algorithms fail to solve the problem (0 number of successful runs) in all 100 independent runs, the average error value obtained by QASMO is better than the average error value obtained by

TABLE 7. Number of successful runs out of 100 independent runs and average number of function evaluations for successful runs for SMO and QASMO (nonscalable problems).

	Number of successful runs (out of 100 runs)		Average number of function evaluations for successful runs	
Problem number	SMO	QASMO	SMO	QASMO
31	100	100	3738	**3492**
32	**98**	49	**430,014**	4,374,449
33	0	0	∼	∼
34	100	100	3585	**3200**
35	100	100	256,584	**55,756**
36	100	100	3304	**3050**
37	**100**	96	**131,394**	180,988
38	100	100	**26,119**	542,972
39	100	100	2009	**1936**
40	100	100	3071	**2913**
41	0	0	∼	∼
42	**100**	96	**24,508**	317,022
43	0	0	∼	∼
44	**100**	99	**13,458**	29,208
45	0	0	∼	∼
46	100	100	10,737	**8753**

Better entries are shown in bold.

TABLE 8. *t*-test results for problems having identical number of successful runs for SMO and QASMO (nonscalable problems).

Problem number	Sign	Problem number	Sign
31	+	38	−
34	+	39	=
35	+	40	+
36	+	46	+

SMO except problem no. 17. Also, it can be seen for problem no. 19 that global optima is obtained by both the algorithms as error value is 0.

Performance index graphs for scalable problems of 30 dimensions have been provided in Figures 10–12. In case (i), the average number of function evaluations for successful runs and the average of error values of 100 runs have been given equal weights. PIs of both algorithms have been displayed in Figure 10. In Figure 10, it can be observed that the value of PI of QASMO is more than that of SMO over the entire range of weight ($0 \leq w \leq 1$). So, in this case, QASMO is a clear winner.

In case (ii), the number of successful runs and the average of error values have equal weights, and the PI graph is given in Figure 11. From the graph, it can be observed that when the weight is less than 0.56 (approx.), QASMO is better than SMO, and when weight is greater than 0.56, SMO outperforms QASMO. The reason for such behavior is that as we go on increasing the weight given to average number of function evaluations for successful runs, the value of PI of QASMO has less value than that of SMO because as we have

TABLE 9. Best, average, and worst of error values obtained in 100 independent runs (nonscalable problems).

	Best		Average		Worst	
Problem number	SMO	QASMO	SMO	QASMO	SMO	QASMO
31	8.53E-08	**1.86E-08**	4.82E-06	**4.38E-06**	9.99E-06	**9.91E-06**
32	**6.79E-08**	2.40E-07	**1.11E-05**	1.39E-04	2.87E-04	**2.87E-04**
33	1.21E-07	**1.04E-08**	4.87E-06	**4.17E-06**	**9.48E-06**	9.92E-06
34	1.74E-07	**3.97E-08**	**4.35E-06**	4.61E-06	9.98E-06	**9.86E-06**
35	1.22E-06	**8.66E-07**	7.24E-06	**6.92E-06**	1.00E-05	**9.96E-06**
36	**9.26E-08**	1.46E-07	4.37E-06	**3.97E-06**	**9.75E-06**	9.91E-06
37	2.41E-06	**1.21E-06**	**7.95E-06**	4.41E-05	**9.99E-06**	9.16E-04
38	1.42E-06	**4.66E-07**	6.25E-06	**6.24E-06**	**9.91E-06**	9.95E-06
39	3.75E-06	3.75E-06	4.99E-06	**4.83E-06**	9.96E-06	**9.94E-06**
40	1.88E-08	**3.74E-09**	4.26E-06	**3.32E-06**	9.94E-06	**9.89E-06**
41	3.75E-04	3.75E-04	**3.88E-04**	5.65E-02	**1.28E-03**	1.52E-01
42	**1.01E-06**	1.55E-06	**5.77E-06**	2.02E-01	**9.88E-06**	5.05E+00
43	5.96E-05	5.96E-05	**5.96E-05**	1.05E-01	**5.96E-05**	5.27E+00
44	**1.74E-08**	4.54E-08	**4.48E-06**	5.36E-02	**9.92E-06**	5.36E+00
45	4.82E-01	4.82E-01	4.82E-01	4.82E-01	4.82E-01	4.82E-01
46	**5.60E-08**	2.20E-07	**3.65E-06**	4.75E-06	9.70E-06	**9.57E-06**

Better entries are shown in bold.

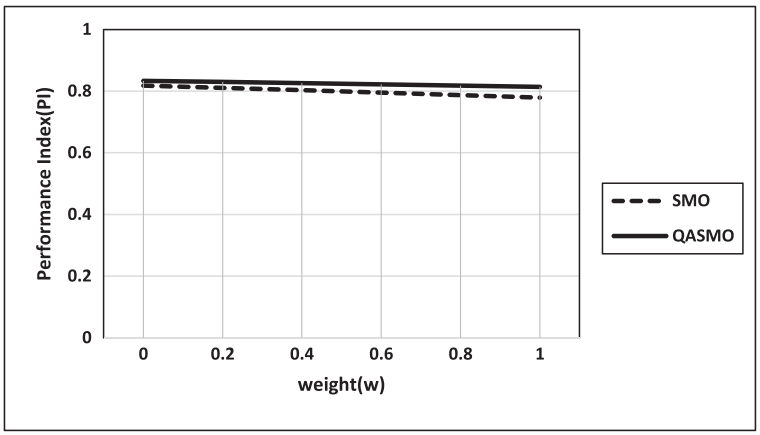

FIGURE 10. Performance index when $k_1 = w$ and $k_2 = k_3 = \frac{1-w}{2}$ (scalable – 30 dimensions).

mentioned previously that although QASMO has more number of successful runs in some cases, but it comes at the cost of more number of function evaluations.

In case (iii), equal weights have been assigned to number of successful runs and average number of function evaluations for successful runs, and the PI graph is displayed in Figure 12. It is clear from the graph that QASMO outperforms SMO in this case.

Also, convergence of the algorithms for some selected problems have been shown in Figures 19 and 20.

Numerical results of scalable problems for 50 dimensions has been discussed in the succeeding discussions.

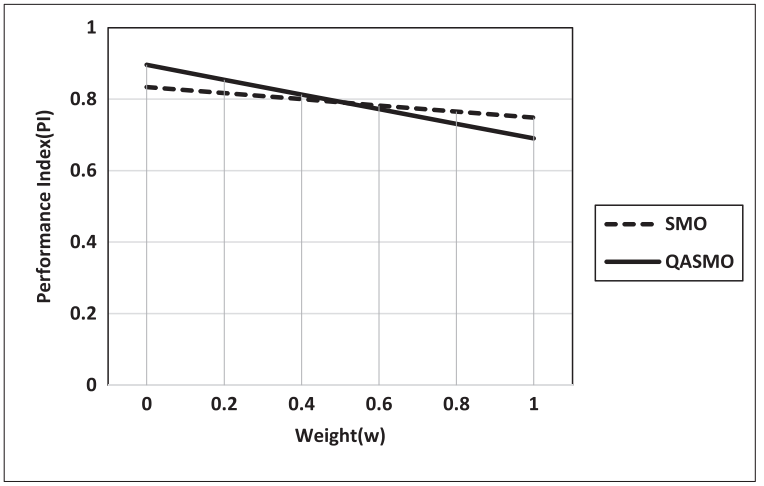

FIGURE 11. Performance index when $k_2 = w$ and $k_1 = k_3 = \frac{1-w}{2}$ (scalable – 30 dimensions).

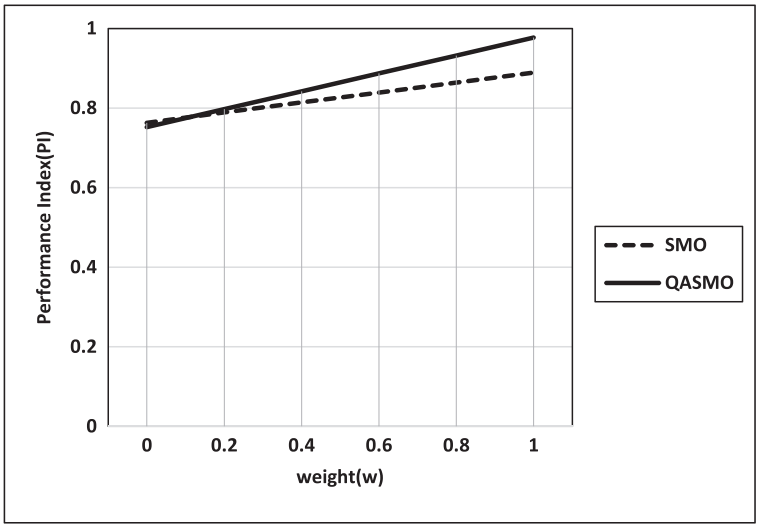

FIGURE 12. Performance index when $k_3 = w$ and $k_1 = k_2 = \frac{1-w}{2}$ (scalable – 30 dimensions).

Table 4 contains information about the number of successful runs and the average number of function evaluations of successful runs. From this table, it can be seen that SMO has 100 successful runs in 21 problems, whereas QASMO has 100 successful runs in 23 problems. In problem nos. 6, 10, and 21, QASMO performs strictly better than SMO according to criterion 1. While, in problem no. 4, SMO performs strictly better than QASMO. Out of 20 problems, where both the algorithms have 100 successful runs, QASMO performs better than SMO over 15 problems according to criterion 1. *t*-test results in Table 5 depict that there is a significant difference in the average number of function evaluations in 16 problems out of 20 problems.

Table 6 provides the best, average, and worst of the error values obtained in 100 runs. There are seven problems (nos. 4, 7, 9, 11, 20, 22, and 24) where QASMO performs strictly

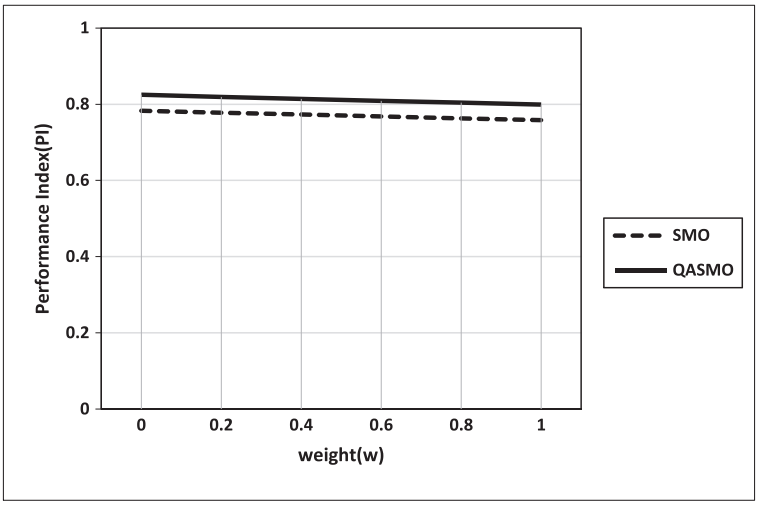

FIGURE 13. Performance index when $k_1 = w$ and $k_2 = k_3 = \frac{(1-w)}{2}$ (scalable – 50 dimensions).

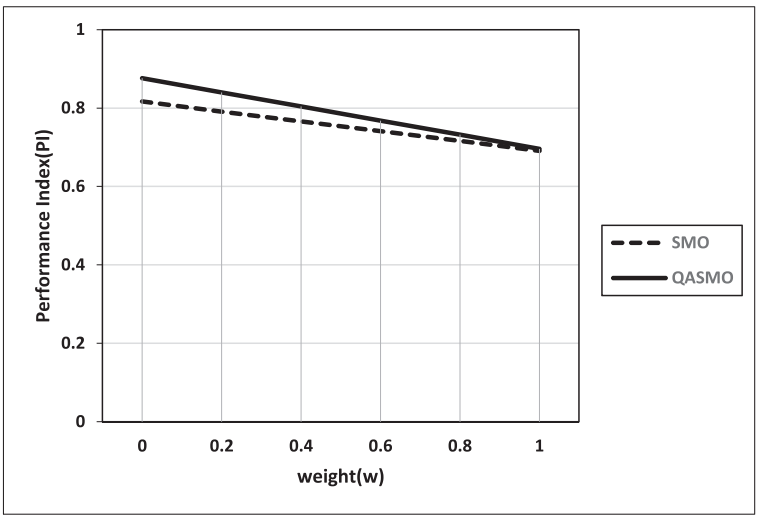

FIGURE 14. Performance index when $k_2 = w$ and $k_1 = k_3 = \frac{(1-w)}{2}$ (scalable – 50 dimensions).

better then SMO according to criterion 2. In problem nos. 6, 12, and 14, QASMO performs better than SMO.

Both the algorithms have the same performance on problem no. 19. PI graphs have been provided in Figures 13–15.

Performance index graph for case (i) has been given in Figure 13, for case (ii), in Figure 14, and for case (iii), in Figure 15. In all the cases, the performance of QASMO is better than SMO.

Also, convergence graph of two problems have been given in Figures 21 and 22.

Hence, from the previous discussion and PI graphs, it can be concluded that QASMO outperforms SMO in most of the scalable benchmark problems considered in the benchmark set using both types of analysis.

Result discussion for nonscalable problems has been given in Tables 7–9.

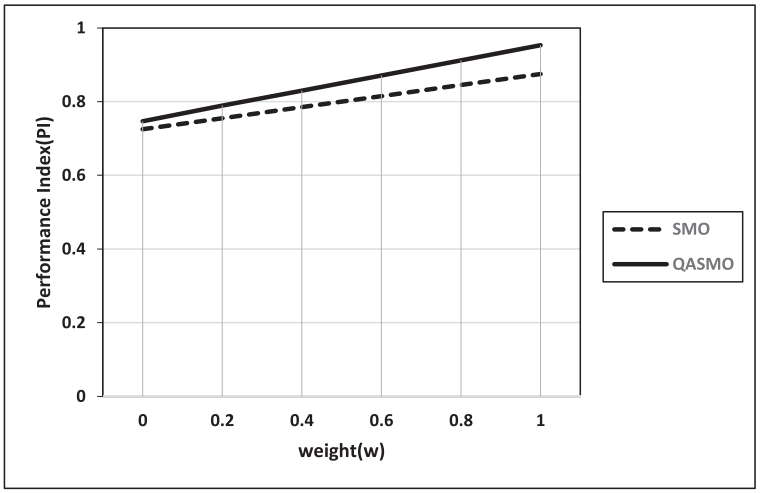

FIGURE 15. Performance index when $k_3 = w$ and $k_1 = k_2 = \frac{(1-w)}{2}$ (scalable – 50 dimensions).

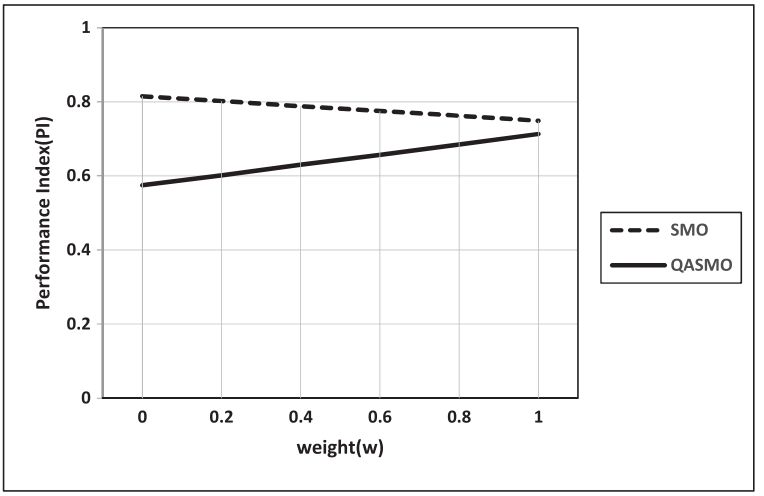

FIGURE 16. Performance index when $k_1 = w$ and $k_2 = k_3 = \frac{(1-w)}{2}$ (nonscalable).

In Table 7, it can be seen that the number of successful runs of SMO is either identical or better than QASMO. SMO has 100 successful runs on 11 problems, and QASMO has 100 successful runs in eight problems. In four problems (nos. 32, 37, 42, and 44), SMO is strictly better than QASMO. Both the algorithms failed to solve problem nos. 33, 41, 43, and 45 in all the runs. QASMO performs better than SMO over seven problems according to criterion 1.

Table 8 contains *t*-test results of the average number of function evaluations. It can be seen from the table that out of 8 problems, where both the algorithms have identical number of successful runs, QASMO performs significantly better than SMO over six problems.

Best, average, and worst of the error values obtained by both algorithms have been listed in Table 9. For best values, QASMO is better than SMO over seven problems, both algorithms have the same best error values in four problems, and SMO is better than QASMO

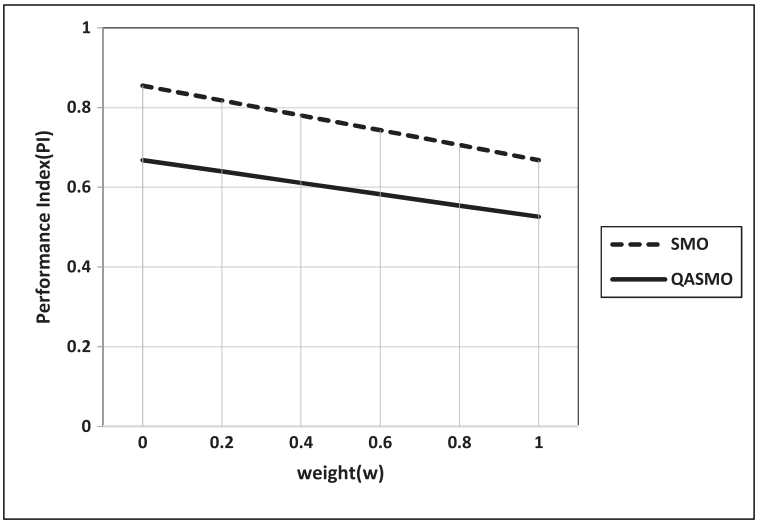

FIGURE 17. Performance index when $k_2 = w$ and $k_1 = k_3 = \frac{(1-w)}{2}$ (nonscalable).

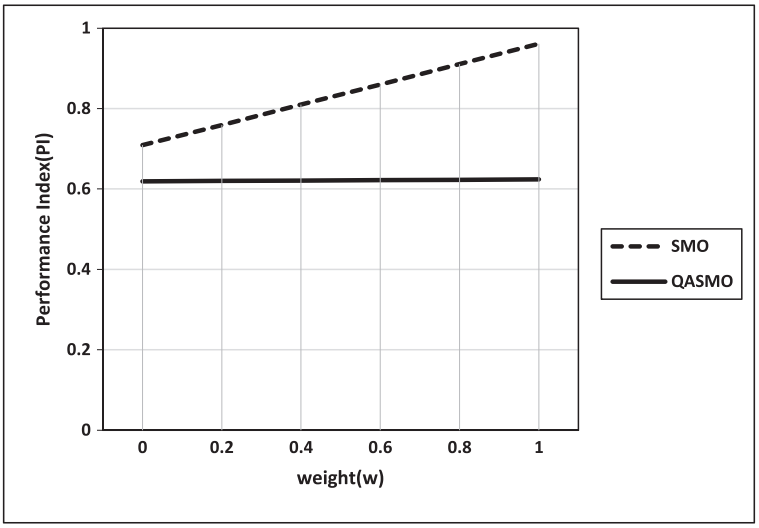

FIGURE 18. Performance index when $k_3 = w$ and $k_1 = k_2 = \frac{(1-w)}{2}$ (nonscalable).

over five problems. For average values, QASMO is better than SMO over 7 problems, same error values for 1 problem and SMO is better than QASMO over 8 problems. For worst values, QASMO outperforms SMO over six problems, and SMO outperforms QASMO over eight problems. Both the algorithms have the same worst error values in two problems. In problem nos. 31, 35, and 40, QASMO performs strictly better than SMO. *t*-test results have been provided in Table 8.

In Tables 7 and 9, it can be concluded that out of four problems (nos. 33, 41, 43, and 45), where both SMO and QASMO has 0 successful runs, the average error value obtained by SMO is better than that obtained by QASMO in two problems (nos. 41 and 43). No clear conclusion can be drawn about the performance of two algorithms on nonscalable problems considered in the benchmark set. In some cases, SMO is performing better, while in other,

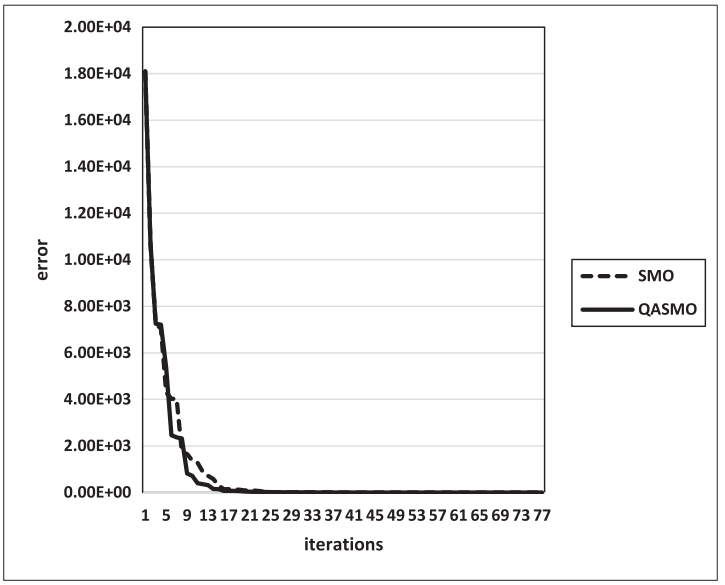

FIGURE 19. Convergence graph of problem no. 2 (30 dimensions).

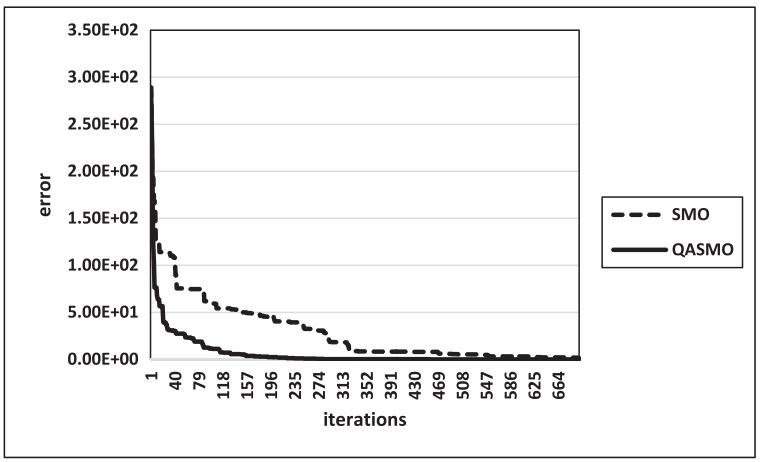

FIGURE 20. Convergence graph of problem no. 11 (30 dimensions).

QASMO is better. PI graphs have been provided in Figures 16–18. In all three cases, the performance of SMO is better than QASMO.

From the earlier discussion on the results of both scalable and nonscalable problems, it can be concluded that although QASMO is showing good performance on scalable problems (listed in benchmark set) in terms of reliability, efficiency, and accuracy as compared with SMO, its performance is reasonable on nonscalable problems.

6. APPLICATION

Determination of molecular confirmation is one of the most challenging problems of computational chemistry, which can be modeled as a global optimization problem. A molecular conformation problem deals with finding the global minimum of a suitable potential

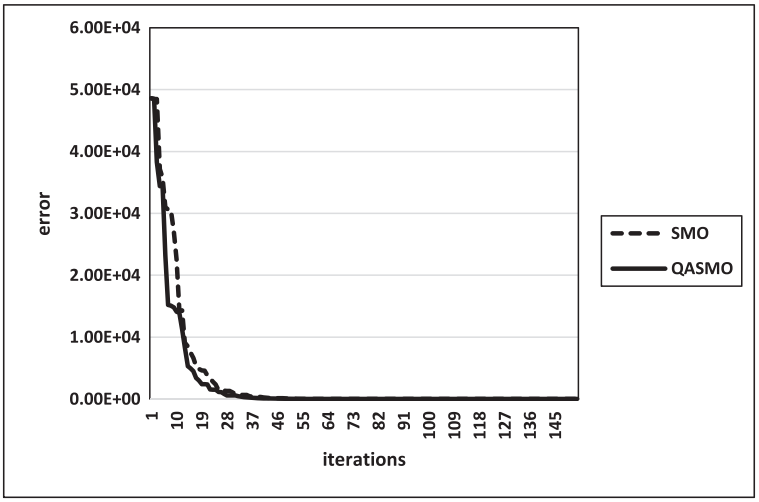

FIGURE 21. Convergence graph of problem no. 2 (50 dimensions).

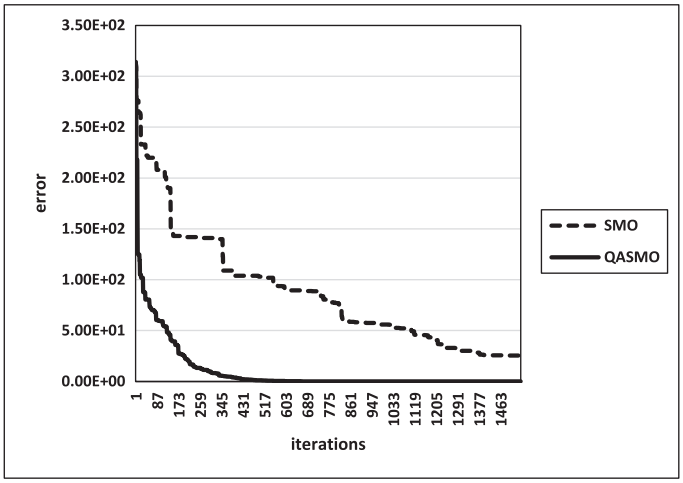

FIGURE 22. Convergence graph of problem no. 11 (50 dimensions).

energy function, which depends on relative positions of atoms. Minimization of this energy provides maximum stability for atomic clusters (Doye 1996). This energy is the sum of several factors including energy caused by the interaction of two nonbonding atoms. Van der Waals interaction is a contributing factor in the energy of interaction between two nonbonding atoms. Van der Waals interaction is characterized by the LJ potential function. The main obstacles in solving LJ problem is the nonlinearity and nonconvexity of the objective function and exponentially increasing number of local minima with increase in the number of dimensions. Despite these difficulties, solution of this problem is very important to facilitate drug design, synthesis, and utilization of pharmaceutical products. Thus, in the past few years, this problem has attracted several researchers from the field of global optimization to apply their algorithms to solve this problem. In Wille and Vennik (1985), it has been shown that complexity of determining global minimum energy of the LJ cluster makes this

TABLE 10. Dimension and search space for different number of atoms.

Number of atoms	Dimension	Search space
3	9	$[-0.52, 0.45]^9$
4	12	$[-0.52, 0.62]^{12}$
5	15	$[-0.75, 0.75]^{15}$
6	18	$[-0.75, 0.75]^{18}$
7	21	$[-0.96, 0.87]^{21}$
8	24	$[-0.9, 1.022]^{24}$
9	27	$[-2, 2]^{27}$
10	30	$[-2, 2]^{30}$

TABLE 11. Number of successful runs and average number of function evaluations of successful runs for SMO and QASMO (Lennard–Jones problem).

	Number of successful runs (out of 100 runs)		Average number of function evaluations for successful runs	
Number of atoms	SMO	QASMO	SMO	QASMO
3	100	100	31,486	**22,154**
4	98	**100**	164,639	36,624
5	18	**100**	287,115	50,135
6	100	100	144,074	**57,802**
7	97	**100**	355,033	62,017
8	94	**100**	314,529	140,117
9	58	**99**	717,820	1,030,777
10	80	**100**	612,590	852,578

Better entries are shown in bold.

TABLE 12. *t*-test results for problems having identical number of successful runs for SMO and QASMO (Lennard–Jones problem).

Number of atoms	Sign	Number of atoms	Sign
3	+	6	+

problem fall in the category of NP-hard problems. This observation has been proved as a motivation for applying metaheuristics to this problem because of their success in solving NP-hard problems in the past few decades. In Deep et al. (2011), LJ problem is solved by applying different variants of real-coded genetic algorithms. In Deep and Madhuri (2011), an attempt has been made to solve LJ problem using a variant of PSO named as globally adaptive inertia weight PSO. In this article, LJ problem is solved using SMO and QASMO. Parameter setting is the same as observed for benchmark problems. Search space for the different clusters has been taken from Deep and Madhuri (2011).

TABLE 13. Best, average, and worst of error values obtained in 100 independent runs (Lennard-Jones problem).

	Best		Average		Worst	
Number of atoms	SMO	QISMO	SMO	QISMO	SMO	QISMO
3	**1.56E-07**	6.01E-07	**5.93E-06**	6.22E-06	**9.92E-06**	9.98E-06
4	3.07E-06	**2.33E-06**	9.85E-06	**7.57E-06**	1.61E-04	**1.00E-05**
5	4.87E-06	**1.96E-06**	6.34E-03	**8.02E-06**	1.43E-01	**9.98E-06**
6	7.02E-08	**5.03E-08**	**5.12E-06**	6.50E-06	**9.91E-06**	1.00E-05
7	2.79E-07	**2.04E-07**	9.52E-06	**6.49E-06**	2.39E-04	**1.00E-05**
8	**5.35E-08**	1.40E-07	1.34E-02	**5.03E-06**	9.59E-01	**9.96E-06**
9	2.26E-07	**7.75E-09**	1.86E-01	**9.24E-03**	2.93E+00	**9.24E-01**
10	**9.62E-08**	1.60E-07	1.24E-01	**4.86E-06**	4.11E+00	**9.98E-06**

Better entries are shown in bold.

6.1. LJ problem

A system containing more than one atom, whose Van der Waals interaction can be described by LJ potential is called an LJ cluster. LJ problem deals with finding the relative position of atoms in a cluster in the three-dimensional Euclidean space in such a way that potential energy is minimum.

Given a cluster of n atoms, LJ problem can be defined mathematically as

$$\mathrm{V} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}^{6}} \right)$$

where r_{ij} is the Euclidean distance between two distinct atoms i and j. Each atom is characterized by a three-dimensional vector say (a, b, and c). Thus, dimension of each solution of LJ problem will be $3n$, where n is the number of atoms in the cluster. Here, target is to minimize V. The potential repels two atoms when they come too close to each other, and this behavior requires a special treatment in molecular dynamics simulation.

In this article, microclusters of atoms from three to ten has been considered for experiment. Parameter setting and comparison criterion are the same as described for benchmark problems. Search space for different number of atoms have been provided in Table 10, and the numerical and statistical results are provided in Tables 11–13

In Table 11, it can be seen that QASMO has more or equal number of successful runs in comparison with SMO in all the cases. QASMO has 100 successful runs in all the cases except for nine atoms cluster, where the number of successful runs is 99. Out of the eight cases, QASMO performs strictly better than SMO in six cases (for 4, 5, 7–9, and 10 atoms) according to criterion 1. Also, in the case of three and six atoms, where both the algorithms have 100 successful runs, QASMO performs better than SMO.

Table 12 provides t-test results for 3 and 6 number of atoms (where both the algorithms have identical number of successful runs), and it shows that the number of function evaluations of QASMO is significantly less than SMO.

Table 13 provides the best, average, and worst of error values obtained by both the algorithms in 100 independent runs. For best error values, QASMO performs better than SMO in five cases (for 4–7 and 9 atoms). For average error values, QASMO outperforms SMO in six cases (for 4, 5, 7–9, and 10 atoms), and for worst error values, QASMO beats

SMO in six cases (for 4, 7–9, and 10 atoms). There are four cases (for 4, 5, 7, and 9 atoms) where QASMO performs strictly better than SMO according to criterion 2.

Therefore, the aforementioned discussion of results validates the applicability of the proposed algorithm in solving real-life problem. Results indicate that QASMO is efficient in solving a highly computationally complex problem like LJ potential for microclusters (from three to ten atoms) at less computational cost.

7. CONCLUSION AND FUTURE SCOPE

In this article, a modified version of SMO is proposed. Local search ability of SMO is improved by incorporating QA operator in it, which is definitely helpful for SMO to explore the surrounding regions of the current global and local leaders. To check the robustness of the proposed algorithm, its performance has been tested over a benchmark set of 46 problems including both scalable and nonscalable problems. Although results are showing improvement in terms of function evaluations success and quality of the solution attained on scalable problems, it is performing moderately on nonscalable problems. In addition to benchmark problems, LJ potential problem for three to ten atoms cluster is solved using QASMO, and results are compared with SMO. Results are better for QASMO for solving LJ problem. In the future, efforts will be made to further improve the performance of the proposed algorithm. Also, LJ problem for large number of atoms will be solved using the improved versions of SMO. We would like to add here that the proposed work is still in its preliminary stage, and several improvements can be performed in the future.

ACKNOWLEDGMENTS

The first author would like to acknowledge the Ministry of Human Resource Development, Government of India, for the financial support and the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

BANSAL, J. C., H. SHARMA, S. S. JADON, and M. CLERC. 2014. Spider monkey optimization algorithm for numerical optimization. Memetic Computing, **6**(1): 31–47.

CLERC, M. 2012. A method to improve standard PSO. Available at: http://clerc.maurice.free.fr/pso/DesignefficientPSO.pdf. Retrieved January 2012.

DEEP, K., and J. C. BANSAL. 2009. Hybridization of particle swarm optimization with quadratic approximation. OPSEARCH, **46**(1): 3–24.

DEEP, K., and K. N. DAS. 2008. Quadratic approximation based hybrid genetic algorithm for function optimization. Applied Mathematics and Computation, **203**: 86–98.

DEEP, K., and MADHURI. 2011. Application of globally adaptive inertia weight PSO to Lennard–Jones problem. International Conference on Soft Computing for Problem Solving (SocProS 2011), IIT Roorkee. *In* Proceedings in Advances in Intelligent and Soft Computing, Vol. 130, Springer, pp. 31–38.

DEEP, K., SHASHI, and V. K. KATIYAR. 2011. Global optimization of Lennard–Jones potential using newly developed real coded genetic algorithms. *In* International Conference on Communication Systems and Network Technologies, Katra, Jammu, pp. 614–618.

DEEP, K., and M. THAKUR. 2007. A new crossover operator for real coded genetic algorithms. Applied Mathematics and Computation, **188**(1): 895–911.

DORIGO, M. 1992. Optimization, learning and natural algorithms, PhD thesis, Politecnico di Milano, Milan, Italy.

DOYE, P. K. 1996. The structure: thermodynamics and dynamics of atomic clusters. Department of Chemistry, University of Cambridge: Cambridge, UK.

HANSEN, N., and A. OSTERMEIER. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. *In* Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, pp. 312–317.

HOLLAND, J. H. 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press: Ann Arbor.

KARABOGA, D. 2005. An idea based on honeybee swarm for numerical optimization. Technical Report, TR06, Engineering Faculty, Computer Engineering Department, Erciyes University, Turkey.

KENNEDY, J., and R. EBERHART. 1995. Particle swarm optimization. Proceedings, IEEE International Conference, **4**: 1942–1948.

KUMAR, S., V. K. SHARMA, and R. KUMARI. 2014. Modified position update in spider monkey optimization algorithm. International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS), **7**(2): 198–204.

LIU, R., L. WANG, W. MA, C. MU, and L. JIAO. 2014. Quadratic interpolation based orthogonal learning particle swarm optimization algorithm. Natural Computing, **13**(1): 17–37.

MOHAN, C., and K. SHANKER (NOW DEEP). 1994. A controlled random search technique for global optimization using quadratic approximation. Asia-Pacific Journal of Operational Research, **11**: 93–101.

PANT, M., M. ALI, and V. P. SINGH. 2009. Differential evolution using quadratic interpolation for initializing the population. *In* Proceedings of the 2009 IEEE International Advance Computing Conference (IACC), Patiala, India, pp. 375–380.

PANT, M., T. RADHA, and V. P. SINGH. 2007. A new particle swarm optimization with quadratic interpolation. International Conference on Computational Intelligence and Multimedia Applications, **1**: 55–60.

PASSINO, K. M. 2002. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Magazine, **22**(3): 52–67.

PRICE, W. L., and G. P. SZEGO. 1978. Towards global optimization. North Holland: Amsterdam, pp. 71–84.

SIMON, D. 2008. Biogeography-based optimization. IEEE Transactions on Evolutionary Computation, **12**(6): 702–713.

STORN, R., and K. PRICE. 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, **11**(4): 341–359.

WILLE, L. T., and J. VENNIK. 1985. Computational complexity of the ground-state determination of atomic clusters. Journal of Physics A: Mathematical and General, **18**: L419–L422.

WOLPERT, D. H., and W. G. MACREADY. 1997. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, **1**: 67–82.

YANG, X. S. 2009. Music-inspired harmony search algorithm. Springer: Berlin Heidelberg, pp. 1–14.

APPENDIX

TABLE A1. Scalable problems.

Test problems	Objective function	Search range	Optimal value
Sphere function	$f_1(x) = \sum_{i=1}^{D} x_i^2$	[−5.12,5.12]	0
De Jong's F4	$f_2(x) = \sum_{i=1}^{D} i x_i^4$	[−5.12,5.12]	0
Griewank	$f_3(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[−600,600]	0
Rosenbrock	$f_4(x) = \sum_{i=1}^{D} \left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right]$	[−100,100]	0
Rastrigin	$f_5(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	[−5.12,5.12]	0
Ackley	$f_6(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ $-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	[−30,30]	0
Alpine	$f_7(x) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	[−10,10]	0
Michalewicz	$f_8(x) = -\sum_{i=1}^{D} \sin(x_i)\left[\frac{\sin(i x_i^2)}{\pi}\right]^{20}$	[0,π]	−9.66015
Cosine mixture	$f_9(x) = \sum_{i=1}^{D} x_i^2 - 0.1\sum_{i=1}^{D} \cos?(5\pi x_i)$	[−1,1]	$-D * 0.1$
Exponential	$f_{10}(x) = -\exp\left(-0.5\sum_{i=1}^{D} x_i^2\right)$	[−1,1]	−1
Zakharov	$f_{11}(x) = \sum_{i=1}^{D} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{D} i x_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{D} i x_i\right)^4$	[−5.12,5.12]	0
Cigar	$f_{12}(x) = x_1^2 + 100000\sum_{i=2}^{D} x_i^2$	[−10,10]	0
Brown3	$f_{13}(x) = \sum_{i=1}^{D-1}\left[\left(x_i^2\right)^{\left(x_{i+1}^2+1\right)} + \left(x_{i+1}^2\right)^{\left(x_i^2+1\right)}\right]$	[−1,4]	0
Schewel prob 3	$f_{14}(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	[−10,10]	0
Salomon problem	$f_{15}(x) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$	[−100,100]	0
Axis parallel hyperellipsoid	$f_{16}(x) = \sum_{i=1}^{D} i x_i^2$	[−5.12,5.12]	0
Pathological	$f_{17}(x) = \sum_{i=1}^{D-1}\left[0.5 + \frac{\sin^2\sqrt{100x_i^2 + x_{i+1}^2} - 0.5}{1 + 0.001\left(x_i^2 + x_{i+1}^2 - 2x_i x_{i+1}\right)^2}\right]$	[−100,100]	0
Sum of different powers	$f_{18}(x) = \sum_{i=1}^{D} \lvert x_i \rvert^i$	[−1,1]	0
Step function	$f_{19}(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	[−100,100]	0
Quartic function	$f_{20}(x) = \sum_{i=1}^{D} i x_i^4 + random[0,1)$	[−1.28,1.28]	0
Inverted cosine wave function	$f_{21}(x) = -\sum_{i=1}^{D-1} \exp\left(-\left(\frac{x_i^2 + x_{i+1}^2 + 0.5 x_i x_{i+1}}{8}\right)\right) *$ $\cos\left(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5 x_i x_{i+1}}\right)$	[−5,5]	$-D+1$
Neumaier3 problem	$f_{22}(x) = \left\lvert \sum_{i=1}^{D} (x_i - 1)^2 - \sum_{i=1}^{D} x_i x_{i+1} \right\rvert$	[−900, 900]	0
Rotated hyper ellipsoid function	$f_{23}(x) = \sum_{i=1}^{D} x_i^2$	[−65.536, 65.536]	0
Levi Montalvo 1	$f_{24}(x) = \frac{\pi}{D}[10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2$ $(1 + 10\sin^2(\pi y_{i+1})) + (y_D - 1)^2]$, where $y_i = 1 + \frac{1}{4}(x_i + 1)$	[−10,10]	0
Levi Montalvo 2	$f_{25}(x) = 0.1\left(\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}\left[(x_i - 1)^2\right.\right.$ $\left.\left.(1 + \sin^2(3\pi x_{i+1}))\right]\right) + (x_D - 1)^2(1 + \sin^2(2\pi x_D))$	[−5,5]	0
Ellipsoidal	$f_{26}(x) = \sum_{i=1}^{D} (x_i - i)^2$	[−D, D]	0
Shifted Parabola CEC 2005	$f_{27}(x) = \sum_{i=1}^{D} z_i^2 + f_{\text{bias}}$, z=(x-o), x=[$x_1 x_2, \ldots, x_D$], O=[$o_1 o_2, \ldots, o_D$]	[−100,100]	−450
Shifted Schwefel CEC 2005	$f_{28}(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2 + f_{\text{bias}}$, z=(x-o), x=[$x_1 x_2, \ldots, x_D$], O=[$o_1 o_2, \ldots, o_D$]	[−100,100]	−450
Shifted Greiwank CEC 2005	$f_{29}(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{\text{bias}}$, z=(x-o), x=[$x_1 x_2, \ldots, x_D$], O=[$o_1 o_2, \ldots, o_D$]	[−600,600]	−180
Shifted Ackley CEC 2005	$f_{30}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ $-\exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e + f_{\text{bias}}$, z = (x − o), x=[$x_1 x_2, \ldots, x_D$], O=[$o_1 o_2, \ldots, o_D$]	[−32,32]	−140

TABLE A2. Nonscalable problems.

Test problems	Objective function	Search range	dimension	Optimal value
Goldstein price function	$f_{31}(x) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 4x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \times \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	$[-2,2]$	2	3
Six hump camel function	$f_{32}(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(4x_2^2 - 4\right)x_2^2$	$x_1 \in [-3,3]$ $x_2 \in [-2,2]$	2	−1.0316
Easom's function	$f_{33}(x) = -\cos(x_1)\cos(x_2)\exp\left[-(x_1 - \pi)^2 - (x_2 - \pi)^2\right]$	$[-100,100]$	2	−1
Beale function	$f_{34}(x) = (1.5 - x_1 + x_1x_2)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2$	$[-4.5,4.5]$	2	0
Colville function	$f_{35}(x) = 100\left(x_1 - x_2^2\right)^2 + (1 - x_1)^2 + 90\left(x_4 - x_3^2\right)^2 + (1 - x_3)^2 + 10.1\left((x_2 - 1)^2 + (x_4 - 1)^2\right) + 19.8(x_2 - 1)(x_4 - 1)$	$[-10,10]$	4	0
Branin Rcos function	$f_{36}(x) = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	$x_1 \in [-5,10]$ $x_2 \in [0,15]$	2	0.3979
Kowalik function	$f_{37}(x) = \left[a_i - \frac{x_1\left(b_i^2 + b_ix_2\right)}{b_i^2 + b_ix_3 + x_4}\right]^2$ where a= {0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246} b= {4.0, 2.0, 1.0, 0.5, 0.25, 0.1667, 0.125, 0.1, 0.0833, 0.07143, 0.0625}	$[-5,5]$	4	0.000307486
2D tripod function	$f_{38}(x) = p(x_2)(1 + p(x_1)) + \lvert x_1 + 50p(x_2)(1 - 2p(x_1))\rvert + \lvert x_2 + 50(1 - 2p(x_2))\rvert$ with : $\begin{cases} p(u) = 1 \; if \; sign(u) \geq 0 \\ = 0 \; if \; sign(u) < 0 \end{cases}$	$[-100,100]$	2	0
Shekel foxholes	$f_{39}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - A_{ij})^6}\right]^{-1}$	$[-65.536,65.536]$	2	0.998
Hartman3	$f_{40}(x) = -\sum_{i=1}^{4} \alpha_i \exp\left[-\sum_{j=1}^{3} A_{ij}\left(x_j - P_{ij}\right)^2\right]$	[0,1]	3	−3.86278
Hartman6	$f_{41}(x) = -\sum_{i=1}^{4} \alpha_i \exp\left[-\sum_{j=1}^{6} A_{ij}\left(x_j - P_{ij}\right)^2\right]$	[0,1]	6	−3.32237
Shekel5	$f_{42}(x) = -\sum_{j=1}^{5}\left[\sum_{i=1}^{4}\left(x_i - C_{ij}\right)^2 + \beta_j\right]^{-1}$	[0,10]	4	−10.1532
Shekel7	$f_{43}(x) = -\sum_{j=1}^{7}\left[\sum_{i=1}^{4}\left(x_i - C_{ij}\right)^2 + \beta_j\right]^{-1}$	[0,10]	4	−10.4029
Shekel10	$f_{44}(x) = -\sum_{j=1}^{10}\left[\sum_{i=1}^{4}\left(x_i - C_{ij}\right)^2 + \beta_j\right]^{-1}$	[0,10]	4	−10.5364
Dekkers and Aarts	$f_{45}(x) = 10^5x_1^2 + x_2^2 - \left(x_1^2 + x_2^2\right)^2 + 10^{-5}\left(x_1^2 + x_2^2\right)^4$	$[-20,20]$	2	−24777
Shubert	$f_{46}(x) = -\sum_{i=1}^{5} i\cos\left((i + 1)x_1 + 1\right)\sum_{i=1}^{5} i\cos\left((i + 1)x_{2+1}\right)$	$[-10,10]$	2	−186.7309