Soft Comput (2017) 21:1765–1784
DOI 10.1007/s00500-015-1880-5

METHODOLOGIES AND APPLICATION

Towards the right amount of randomness in quantum-inspired evolutionary algorithms

C. Patvardhan[1] · **Sulabh Bansal**[1,3] · **Anand Srivastav**[2]

Published online: 1 October 2015

Abstract Quantum-inspired evolutionary algorithms (QIEAs) combine the advantages of quantum-inspired bit (Q-bit), representation and operators with evolutionary algorithms for better performance. Using quantum-inspired representation the complete binary search space can be generated by collapsing a single Q-bit string repeatedly. Thus, even a population size of 1 can be taken in a QIEA implementation resulting in enormous saving in computation. Although this is correct in theory, QIEA implementations run into trouble in exploring large search spaces with this approach. The Q-bit string has to be initialized to produce each possible binary string with equal probability and then altered slowly to probabilistically favor generation of strings with better fitness values. This process is unacceptably slow when the search spaces are very large. Many ideas have been reported with EAs/QIEAs for speeding up convergence while ensuring that the algorithm does not get stuck in local optima. In this paper, the possible features are identified and systematically introduced and tested in the QIEA framework in various combinations. Some of these features increase the randomness in the search process for better exploration and the others compensate by local search for better exploitation together enabling a judicious combination tailored for particular problem being solved. This is referred to as "right-sizing the randomness" in the QIEA search. Benchmark instances of the well-known and well-studied Quadratic Knapsack Problem are used to demonstrate how effective these features are—individually and collectively. The new framework, dubbed QIEA-QKP, is shown to be much more effective than canonical QIEA. The framework can be utilized with profit on other problems and is being attempted.

Keywords Evolutionary algorithm · Combinatorial optimization · Quantum-inspired evolutionary algorithm · Quadratic Knapsack Problem

Communicated by V. Loia.

✉ Sulabh Bansal
sulabh.bansal.78@gmail.com

[1] Faculty of Engineering, Dayalbagh Educational Institute, Agra 282010, India

[2] Institut für Informatik, Christian-Albrechts-Universität zu Kiel, 24098 Kiel, Germany

[3] 1/25, Hazuri Bhawan, Peepal Mandi, Agra, India

1 Introduction

Evolutionary algorithm (EA) is a population-based nature-inspired meta-heuristic search technique (Bäck 1996). Quantum-inspired evolutionary algorithms (QIEAs) proposed by Han and Kim (2002, 2004), combine the advantages of quantum-inspired bit (Q-bit) representation and operators with EAs for better performance. QIEAs have been shown to be competitive for several search and optimization problems. Several attempts that utilize QIEAs for the solution of a wide variety of problems have been reported in literature.

The main strengths of the QIEAs are as follows:

(i) QIEAs have better representation power using quantum-inspired bits (Q-bits) to enable use of smaller populations (ideally even a size of 1). Smaller populations require lesser computation during search.
(ii) QIEAs have an Estimation of Distribution Algorithm (EDA) style functioning with implicit determination of distributions leading to better solutions (Platel et al. 2009; Zhang 2011).
(iii) QIEAs provide an extremely flexible framework that can be adapted for the solution of both real—parameter

function optimization problems as well as combinatorial optimization problems. This makes them very versatile in their applicability.
(iv) The QIEA framework also provides flexibility for inclusion of features appropriate for a given problem towards delivering better search performance (Zhang 2011).
(v) QIEAs inherently favor exploration of the search space initially, while gradually shifting towards exploitation as the search progresses, which is a desirable aspect.
(vi) There is a possibility of utilizing one of several termination criteria appropriate for the problem at hand (Han and Kim 2004).

QIEAs are attractive because of the advantages listed above. This has spurred greater recent interest in the QIEAs. The fundamental basis of their working is well understood. QIEAs, of course, are not a "one-size fits all" solution. The No Free lunch Theorem prohibits that. The conspicuous limitations of QIEAs are as follows. Many of these are shared with other EAs as well.

(i) Slow convergence may result from use of small Q-bit rotations. This is because initial configurations of Q-bits result in random sampling and search.
(ii) Use of large Q-bit rotations may cause the algorithm to miss a good solution completely.
(iii) Inclusion of features promoting faster convergence may cause the algorithm to get stuck in local optima.
(iv) Slow convergence may limit the problem sizes that can be tackled using QIEAs.
(v) Implementation of QIEAs, just as other EAs, is more an art to enable balance of computational effort devoted to exploration and exploitation that is required for good search performance on the given problem.

QIEAs only provide a very broad framework. Any attempted implementation, therefore, must make judicious use of the framework and include features suited to the particular problem at hand in order to get the desired performance. The objective in any attempted implementation of QIEA is to balance exploration and exploitation thus achieving convergence to optimal or near optimal solution without requiring prohibitively large computation. This is possible by "right-sizing the randomness" in the QIEA search. This has led to a plethora of attempts reported in the literature that report an "Improved QIEA" for the solution of a particular problem as shown in Table 1 for various knapsack problems. Some of the modifications typically proposed are as follows:

(i) Utilizing a good solution obtained using some heuristic to guide the search.
(ii) Hybridizing the QIEA with some kind of generic local search method to speed up convergence by promoting better exploitation.
(iii) Inclusion of domain knowledge in the search process.
(iv) Incorporation of standard genetic operators like crossover and mutation.
(v) Repair of infeasible solutions generated during the search process with inclusion of some domain-specific knowledge to push towards better solutions.
(vi) Re-initialization/modification of Q-bits strings to get the search out of local minima if the QIEA is stuck.
(vii) Better termination criteria.

Thus, although it is well known that QIEAs are competitive for a large variety of problems, a particular QIEA has to be designed for every problem to achieve high performance that is competitive with respect to the state-of-the-art algorithms for the problem. As listed above, there are many design alternatives available for consideration. That is the strength of the QIEA framework. But, it also means that an implementation that works for a given problem type has to be hand crafted carefully. Some basic principles well understood in the context of EAs may be utilized but, for most part, it is an art.

In this paper, an attempt has been made to systematically analyze the extant approaches to enhance QIEAs to determine how effective they are on large combinatorial optimization problems. An attempt is made to study the various implementations and draw conclusions regarding their relative effectiveness. On the basis of the analysis, an extended framework that incorporates these ideas is provided.

A carefully selected combination of features is incorporated such that some increase the diversity by increasing the randomness and others increase the exploitation of the discovered regions in search space using deterministic heuristic techniques. This is referred to as "right-sizing the randomness" in the QIEA search. The proposed framework is developed and its performance studied using Quadratic Knapsack Problem (QKP) as a vehicle and is dubbed as QIEA-QKP.

QKP is NP-hard in the strong sense and has several reported applications. This has spurred tremendous interest in the problem and it is well-studied in the literature. On several benchmark problems, QIEA-QKP delivers performance which is substantially better than the canonical QIEA on both counts, i.e., solution quality and computation time.

The rest of the paper is organized as follows. The extant QIEAs are analyzed in Sect. 2. The QKP is described in Sect. 3. The proposed framework QIEA-QKP is presented in Sect. 4, along with explanation of the various features incorporated in it. The computational performance of QIEA-QKP along with systematic study of effects of various features incorporated is presented in Sect. 5. Conclusions are presented in Sect. 6.

2 Quantum-inspired evolutionary algorithms (QIEAs)

The QIEAs introduced in Han and Kim (2002) are population-based stochastic evolutionary algorithms. QIEAs use a Q-bit vector, to represent the probabilistic state of individual. Each Q-bit is represented as $q_i = \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}$, α_i, β_i are complex numbers so that $|\alpha_i|^2$is the probability of state being 1 and $|\beta_i|^2$ is the probability of state being 0 such that $|\alpha_i|^2 + |\beta_i|^2 = 1$. For the purpose of QIEAs, α_i and β_i are assumed to be real. Thus, a Q-bit string with n bits represents a superposition of 2^n binary states and provides an extremely compact representation of the entire search space.

The process of generating binary strings from the Q-bit string is known as observation. To observe the Q-bit string a string R consisting of the same number of random numbers between 0 and 1 is generated. Each element P_i is set to 0 if R_i is less than square of Q-bit Q_i and 1 otherwise. In each iteration, several solution strings are generated from Q by observation in this manner and their fitness values are computed. The solution with best fitness is then identified. The updating process moves the elements of Q towards the best solution slightly such that there is a higher probability of generation of solution strings, which are similar to best solution, in subsequent iterations. A quantum-inspired gate (Q-gate) is utilized for this purpose.

One such gate, used by the QIEAs presented in this work, is the Rotation Gate, which updates the Q-bits as follows:

$$\begin{bmatrix} \alpha_i^{t+1} \\ \beta_i^{t+1} \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i^t \\ \beta_i^t \end{bmatrix},$$

where α_i^{t+1} and β_i^{t+1} denote values in ith Q-bit in $(t+1)$th iteration and $\Delta\theta_i$ is equivalent to the step size in typical iterative algorithms in the sense that it defines the rate of movement towards the currently perceived optimum.

The above description outlines the basic elements of a QIEA. Observing a Q-bit string n times yields n different solutions because of the probabilities involved. The fitness of these is computed and the Q-bit string Q is updated towards higher probability of producing strings similar to the one with highest fitness. This sequence of steps continues; these ideas can be easily generalized to work with multiple Q-bit strings.

QIEAs as such are applicable to a large variety of problems, but they may be too slow and computation intensive to be of use in case of large-sized problems. Several attempts have been made in the past to modify QIEAs in order to solve hard problems. Table 1 lists some of the attempts made to solve variants of KP using QIEA or modified QIEAs. Very few attempts have been made to solve really difficult instances of KP or its difficult variants like MoKP and QKP. Most of the attempts are targeted to solve the simplest form of KP (i.e., 0/1 Knapsack Problem) instances and that too having very small size.

```
    Procedure SQIEA
1   t ← 0;
2   Initialize (Q(t)); Initialize b;
3   Make P(t) from Q(t);
4   Repair P(t);
5   copy P(t) to B(t);
6   while ( t<MaxIterations)
7   {
8     t ← t+1;
9     for r from 0 to η1 do{
10          for s from 0 to η2 do{
11            Make P(s) from Q(t);
12            Repair P(s);
13            Evaluate P(s);
14            for each j ∈ {1, ..., n} if ( p_j^s better than p_j^t )  p_j^t ← p_j^s;
15          } //*for s*//
16          for each j ∈ {1, ..., n} if ( p_j^t is better than b_j^t)  b_j^t ← p_j^t;
17          for each j ∈ {1, ..., n}if (b_j^t is better than b) b ← b_j^t ;
18          for each j ∈ {1, ..., n} Update q_j^t based on b_j^t;
19    } //*for r*//
20  for each j ∈ {1, ..., n}Update q_j^t based on b;
21  } //*while*//
```

Fig. 1 Pseudo-code for SQIEA

The simple QIEA (SQIEA) is illustrated in Fig. 1 which uses the following notations.

Notations:

$Q(t)$	Q-bit population in tth iteration.
$P(t)$	population of binary solutions in tth iteration.
$B(t)$	population of best solutions in tth iteration.
q_j^t	jth individual in $Q(t)$.
p_j^t	jth individual in $P(t)$.
b_j^t	jth individual in $B(t)$.
b	best solution observed so far.
MaxIterations	maximum number of iteration set as termination criterion by the user.
η_1	number of times a local update of Q-bits is performed before updating based on global best solution found so far.
η_2	number of multiple observations made on a Q-bit individual before its local update.
n	number of individuals in the population.
t	the current iteration.

The steps of SQIEA are explained in brief in the following:

- Q-bits in $Q(t)$ are initialized with $1/\sqrt{2}$ and best binary solution b with zeroes (lines 1 and 2). Before starting iterations, individuals of $Q(t)$ are observed into $P(t)$ and these solutions are repaired to make them feasible (lines 3 and 4). $B(t)$ is initialized by these feasible solutions (line

Table 1 Various modifications those have been considered in QIEAs to solve variants of KP

Modifications	Examples	Type	Maximum size n = items count, m = knapsacks count
Original QIEA (QIEA-o)	Han and Kim (2002, 2003)	KP	$n = 500$
Modified initialization of Q-bit individuals in QIEA	Han and Kim (2004)	KP	$n = 500$
	Zhao et al. (2006)	KP	$n = 500$
	Zhang et al. (2008)	KP	$n = 600$
	Tayarani and Akbarzadeh (2008)	KP	$n = 500$
	Patvardhan et al. (2014a)	KP	$n = 290,000$
	Patvardhan et al. (2014b)	MKP	$n = 10,000, m = 10$
Modification in termination criteria	Han and Kim (2004)	KP	$n = 500$
Modifications in update procedure, e.g., choice of attractor, different Q-gate, other learning strategies, etc.	Han and Kim (2004)	KP	$n = 500$
	Platel et al. (2007)	KP	$n = 500$
	Patvardhan et al. (2007)	DKP	$n = 10,000$
	Zhang et al. (2008)	KP	$n = 600$
	Li et al. (2009)	MoKP	$n = 750, m = 4$
	Zhang et al. (2012)	KP	$n = 40,000$
	Qin et al. (2012)	KP	$n = 3000$
Modifying repair function based on domain knowledge	Zhao et al. (2006)	KP	$n = 500$
	Patvardhan et al. (2014a)	KP	$n = 29,0000$
	Patvardhan et al. (2014b)	MoKP	$n = 10,000, m = 10$
Replacing local and/or global migration of QIEA-o with other strategies	Mahdabi et al. (2008)	KP	$n = 500$
	Imabeppu et al. (2008)	KP	$n = 500$
	Patvardhan et al. (2014a)	KP	$n = 290,000$
	Patvardhan et al. (2014b)	MoKP	$n = 10,000, m = 10$
Incorporation of genetic operator mutation	Patvardhan et al. (2007)	DKP	$n = 10,000$
	Patvardhan et al. (2012)	QKP	$n = 200$
	Patvardhan et al. (2014a)	KP	$n = 290,000$
	Patvardhan et al. (2014b)	MKP	$n = 10,000, m = 10$
Changing size of population of Q-bit individuals	Lu and Yu (2013)	MoKP	$n = 750, m = 4$
Re-initialization of Q-bits	Mahdabi et al. (2008)	KP	$n = 500$
	Patvardhan et al. (2014a)		$n = 290,000$
	Patvardhan et al. (2014b)		$n = 10,000, m = 10$
Inclusion of domain knowledge in the search process	Kim et al. (2006)	MoKP	$n = 750, m = 2$
	Lu and Yu (2013)	MoKP	$n = 750, m = 4$
	Patvardhan et al. (2007)	DKP	$n = 10,000$
	Patvardhan et al. (2014a)	KP	$n = 290,000$
	Patvardhan et al. (2014b)	MKP	$n = 10,000, m = 10$
Parallel implementation	Han et al. (2001)	KP	$n = 500$
	Nowotniak and Kucharski (2012)	KP	$n = 250$

KP 0/1 Knapsack Problem, *MKP* Multiple Knapsack Problem, *MoKP* Multi-objective Knapsack Problem, *QKP* Quadratic Knapsack Problem, *DKP* Difficult 0/1 Knapsack Problem

5). The SQIEA iterates MaxIterations times through the tasks described in lines 6–21. The following tasks are repeated η_1 times first (lines 10–19):

- The Q-bits in $Q(t)$ are collapsed and repaired η_2 times to form feasible solutions in $P(s)$ and the corresponding best solutions are retained in $P(t)$ (lines 11–15).
- Individual at every position in $B(t)$ is replaced by corresponding individual in $P(t)$ if found better (line 16).
- The best solution from among b and $B(t)$ is moved into b (line 17).
- Update (local): each individual in $Q(t)$ is updated based on the corresponding best solution in $B(t)$ (line 18).

- The individuals in $Q(t)$ are updated (global) based on the best solution observed so far. Each iteration is started after updating globally (line 20).

3 The Quadratic Knapsack Problem (QKP)

The 0/1 Quadratic Knapsack Problem (QKP) is a generalization of the 0/1 Knapsack Problem (KP) introduced by Gallo (1980). Given n items to be filled in a knapsack where w_j is the positive integer weight of jth item, c is a positive integer knapsack capacity and an $n \times n$ nonnegative integer matrix $P = (p_{ij})$ is given, where p_{jj} is a profit achieved if item j is selected, and, for $j > i$, $p_{ij} + p_{ji}$ is the additional profit achieved if both items i and j are selected. QKP is to find a subset of items whose total weight is not more than knapsack capacity c such that the overall profit is maximized. If x_j is a binary variable which is equal to 1 if jth item is selected and 0 otherwise, the problem is formulated as follows:

$$\begin{aligned} \text{Maximize:} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} x_i x_j \\ \text{Subject to:} \quad & \sum_{j=1}^{n} w_j x_j \leq C \\ & x_j \in \{0, 1\}, \quad j \in \{1, \ldots, n\}. \end{aligned} \tag{1}$$

Sometimes matrix P is considered symmetric such that $p_{ij} = p_{ji}$ for all i and j. In such a problem, additional profit achieved if both items i and j are selected is considered as p_{ij} rather than $p_{ij} + p_{ji}$, for $j > i$. Thus, some researchers formulate the problem as follows:

$$\begin{aligned} \text{Maximize:} \quad & \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij} x_i x_j \\ \text{Subject to:} \quad & \sum_{j=1}^{n} w_j x_j \leq C \\ & x_j \in \{0, 1\}, \quad j \in \{1, \ldots, n\}. \end{aligned} \tag{2}$$

The KP is a particular case of QKP which arises when $p_{ij} = 0$ for all $i \neq j$. *Clique* problem, is also a particular case of QKP, which calls for checking whether, for a given integer k, a given undirected graph $G = (V, E)$ contains a complete subgraph on k nodes. The popular optimization version of *Clique*, called *Max Clique*, calls for an induced complete subgraph with a maximum number of nodes. The *Max Clique*, can be solved using a QKP algorithm by using binary search.

Max Clique is not only NP-hard in stronger sense, but is one of the hardest combinatorial optimization problems. Same properties apply to QKP as well. Pseudo-polynomial time algorithms exist for KP but none for QKP. QKP is considered much more difficult than simple KP (Caprara et al. 1999; Pisinger 2007).

Due to the generality and complexity of QKP, it is studied widely and shown to have applicability in several areas like facility location problems , compiler design (Johnson et al. 1993), finance, VLSI design (Ferreira et al. 1996) and weighted maximum b-clique problem (Dijkhuijen and Faigle 1993; Park et al. 1996).

Ever since Gallo (1980) introduced QKP and presented method to derive the upper bound using upper planes several attempts have been made in past to solve QKP. The existing exact algorithms suffer from high time complexity (Pisinger 2007; Kellerer et al. 2004).

Julstorm (2005) presented a greedy Genetic Algorithm (GGA) to solve some benchmark instances given by Billionet and Soutif (2004) (BS benchmark instances) with upto 200 variables to optimality in 902 out of 1000 trials using around 15000 function evaluations (FEs) on an average per trial. A Mini-Swarm algorithm is proposed by Xie and Liu (2007) is shown to solve all BS benchmark instances with 100 and 200 binary variables to optimality with high probability in a reasonable time. Patvardhan et al. (2012) presented known best QIEA for QKP (dubbed QIEA-PPA in this paper). They compared their results with Naïve EA and greedy EA by Julstorm (2005) and showed better performance in terms of consistency in finding the optimal solution.

4 QIEA-QKP

As stated above, SQIEA provides a broad framework with scope for enhancements. Here, the modified algorithm designed for QKP is presented (dubbed QIEA-QKP). The enhancements which have been included to control the randomness effectively in QIEA-QKP are as follows:

(i) Initializing Best Solution "b" with a good solution obtained using a heuristic.
(ii) Using the sort orders of items in a problem as domain knowledge during following operations.
   a. Initializing the Q-bit individuals to depict better estimations of distribution models.
   b. Fortifying the repair function to improve quality of solutions.
(iii) Improving the local best solutions using combination of randomized and non-randomized heuristic-based local search.
(iv) Corrective step when the algorithm appears to be stuck in a local optimum.
(v) Re-initialization of Q-bit individuals.
(vi) Replacing the non-performing Q-bit individuals by best.

```
    Function SolveGreedy() /* returns GreedySolution*/
1   GS ← {1,...,n}, w←∑_{i=0}^{n} w_i;
2   forever do {
3      Let i | min_{i ∈ GS}(RVD_i^{GS});
4      if (w > C) { GS ← GS / i; w ← w- w_i;}
5      else { ImproveLocal(GS); return (GS);}
6   } //* forever *//
```

Fig. 2 Pseudo-code for SolveGreedy

```
    Function SortGreedy() /*returns GreedyOrder*/
1   PS ← Φ;j←1;
2   Let i | max_{0≤i≤n}( p_ii/w_i);
3   GO[j] ← i;
4   PS ← PS ∪ i;
5   for j from 2 to n{
6      Let i | max_{i ∉ PS} RVD_i^{PS}; GO[j] ← i; PS ← PS ∪ i;
7   }//* for j *//
8   return(GO);
```

Fig. 3 Pseudo-code for SortGreedy

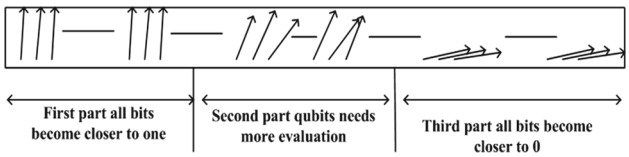

Fig. 4 Initialization of Q-bits. Q-bits for items shown are sorted in GreedyOrder from *left* to *right*

4.1 Initializing best solution "*b*" with a good solution obtained using a heuristic

The "update" procedure modifies Q-bit individuals during evolution so that they favor generation of solutions closer to best solutions obtained so far. Initializing the best solution with a better solution (rather than any random solution) ensures that the better distribution models are found earlier. The greedy solution is a fairly good solution and the QIEA-QKP initializes the best solution with it. A greedy solution can be formed for QKP relatively quickly using a simple constructive heuristic. Starting with an infeasible solution which has all the items included in the knapsack the greedy algorithm removes items iteratively till the solution becomes feasible. Each time an item having minimum relative profit density is removed from solution. Let P represent a partial solution such that $i \in P$ iff ith item is included in the knapsack. The relative value density of an item i with respect to P (RVD_i^P) is computed as $(p_{ii} + \sum_{j \in P/\{i\}} p_{ij})/w_i$. Let GS be a set representing the solution. The greedy heuristic used in QIEA-QKP (dubbed SolveGreedy) is depicted in Fig. 2.

4.2 Using heuristic-based sort orders as domain knowledge

Orderings of the elements named GreedyOrders, which provide knowledge about their priority for inclusion in the knapsack, are computed. These are used to initialize the Q-bit individuals (Sect. 4.2.1) and also while repairing infeasible solutions (Sect. 4.2.2).

Starting with an empty solution, elements are added iteratively. During each iteration, the item having maximum RVD with respect to the partial solution available before the iteration is added. The process is depicted in Fig. 3. The item having the maximum diagonal profit can be selected as the first item with the rest being selected as stated. The GreedyOrder so generated may not provide the optimal sequence of items to be selected. Different choice of the first item in step 3 leads to a different order. Thus, multiple orders are generated by taking different items as the first item instead of item 1.

Initializing Q-bits or repairing the solutions based on a single GreedyOrder, makes the search explore a specific region of solution space with higher probability. Usage of multiple GreedyOrders enables exploring different regions simultaneously. All these regions converge towards local best solutions produced in the particular region with subsequent movement towards global best solution. Such an arrangement promotes judicious exploration and exploitation of the favorable regions.

4.2.1 Initializing the Q-bit Individuals to depict the better estimations of distribution models

The Q-bit individuals are initialized based on the GreedyOrders as shown in Fig. 4. The items are divided in 3-parts based on where they lie in order they being preferred: first part contains items highly preferred for selection in knapsack, items in second part have medium priority and items in third part have low priority. Hence, Q-bits for items lying in first part (third part) are assigned values closer to 1 (0) so that they have high (low) probability of collapsing to value 1. Q-bits for items lying in second part necessarily require more decision making to converge to either 0 or 1. Hence values between 0 and 1 are assigned to them. As a result, QIEA-QKP starts exploiting the favorable area or region in search space represented by such initialized Q-bit individuals. Multiple GreedyOrders are used in round-robin fashion to help initialize all the Q-bit individuals in different manner to target different favorable regions in search space.

4.2.2 Fortifying the repair function to improve quality of solutions

SQIEA uses a simple "*repair*" function after it observes the Q-bits through the "*make*" procedure to make the observed

```
      Procedure RepairGreedy (x, GO)
1       knapsack-overfilled ← false;
2       w ← Σ_{j=1}^{n} w_j x_j;
3       if (w > C) knapsack-overfilled ← true;
4       while (knapsack-overfilled) {
5         for i from GO[n] to GO[1]
6           { if (x_i=1) {x_i ← 0; w← w – w_i;} }
7         if (w ≤ C) knapsack-overfilled ← false;
8       }//*while*//
9       for j from GO[1] to GO[n] {
10        if ((x_j=0) and (w+w_j ≤ C)){ x_j ←1; w← w +w_j;}
11      } //*for j*//
```

Fig. 5 Pseudo-code for RepairGreedy

solution feasible. In QIEA-QKP, the repair function is fortified to improve the quality of solutions while making them feasible. This improves the speed of convergence. A *Greedy-Order*, as explained above, determines the relative change in profit when an item is included or excluded from the knapsack. Accordingly, in each repair step, items closest to the end of a GreedyOrder are removed and items closest to the beginning are added as necessary. The pseudo-code is given in Fig. 5. Array GO, provides items in the GreedyOrder sequence.

If the same order is used each time in repairing all the infeasible solutions, the results would be almost alike every time. Thus, multiple orders are used in repair function in a round-robin manner. Such a policy helps in maintaining diversity in the solutions generated by the repair function.

4.3 Improving the local best solutions

The local best solutions are further improved using a local "improve" function also referred in Sect. 4.1. It tries to explore the quality of all solutions in vicinity of a local best solution and selects the best. It iteratively executes passes as long as gain in profit is observed. The ith element (if not in solution) is either *included* or *replaced* by an item j already in solution after each pass. The action of inclusion (or replacement) is performed for that i (or pair i and j) which results in maximum gain in overall profit of the solution. Let P be a feasible solution, the pseudo-code of procedure used to improve profit of P is given in Fig. 6. The function defined in Fig. 6 is computationally expensive. Moreover, it tries to convert worse solutions to locally best solution and thus increases the chance of generating same solution many times. So a lighter version, RandImproveLocal, is also implemented where the forever loop in step 1 of ImproveLocal is executed a fixed number of times instead, each time randomly picking an element from a list of not included items in step 3. This lighter version is performed on half of the individuals to save computation.

```
      Procedure ImproveLocal(P)
1     forever do{
2       bg←0;m_i=-1;m_j=-1;
3       for ∀i ∉ P {
4        if (w_k ≤ C − Σ_{k∈P} w_k) {
5            g = p_ii + Σ_{k∈P}(p_k p_i);
6           if (g > bg) { bg ← g;m_i=i;}
7        }//* if*//
8        else{
9          for ∀j ∈ P do
10           g = p_ii − p_jj + Σ_{k∈P/j}(p_k p_i − p_k p_j);
11           if (g > bg){ bg ← g; m_i=i;m_j=j;}
12         }//*for j*//
13       }//*else*//
14      }//*for i*//
15      if (bg = 0) exit;
16      P ← P ∪ m_i; if(m_j≠-1) P ← P/ m_j;
17    } //* forever *//
```

Fig. 6 Pseudo-code for ImproveLocal

4.4 Mutation of solutions stuck in local optimum

EAs suffer from tendency of getting stuck in local optima. All the modifications described above help the algorithm to exploit the search space around the greedy solutions increasing the speed of convergence, but they also increase the tendency to get stuck in local optima. To combat this problem, if a new solution generated is seen to be close to global best solution found so far it is mutated. During mutation 2–3 bits in the solution vector are randomly selected and changed to 0. Elements with better RPD are then iteratively included in solution as long as the solution remains feasible. To check closeness of two solutions, Hamming distance between them is calculated. This operator improves diversity without increasing the computational effort. It helps to explore the solution space around a current solution such that local optimal in vicinity is not missed. This improves the chances of finding optimal in case it is in vicinity of the converging solution.

4.5 Re-initialization of Q-bit individuals

All the solutions generated from a Q-bit individual may still be same after a sequence of generations even though mutation is employed. It clearly indicates that the Q-bit individual has converged and further observations made on it will not yield any better solutions. So instead of wasting cycles on such a Q-bit string, it is re-initialized to restore diversity. But, reinitializing it again in same way as described in Sect. 4.1 may not make any difference. Thus, each Q-bit in individuals which generate same solution for more that 3 times out of 5 is set to $1/\sqrt{2}$.

4.6 Replacing the non-performing Q-bit individuals by best individual found so far (StochasticPurge)

The Q-bit individuals are assigned a fixed lifetime to perform. If a Q-bit individual is found to perform worse than average after the defined lifetime, there remains 50 % chance for it to survive. In case it ceases to exist, it is replaced by the best performer found so far.

The complete pseudo-code for QIEA-QKP is presented in Fig. 7. As in pseudo-code for SQIEA, $Q(t)$ is the Q-bit population after tth iteration; $P(t)$ is the population of individual solutions; $B(t)$ is the set of best solutions corresponding to each individual; C is the capacity of the knapsack. Individuals in $Q(t)$, $P(t)$ and $B(t)$ are referred by q_j^t,p_j^t and b_j^t, respectively, for each $j \in n$; b refers to global best solution and bqbit refers to Q-bit individual which generated the best solution; be and we refer to best and worst solutions evolved so far through out the evolution. In the following, a brief description is given of the procedures for which pseudo-code has not been given already.

- *MultipleSortGreedy* (GreedyOrders, N) A list of different sort orders, GreedyOrders, of length N is generated. All sort orders are generated using the SortGreedy() procedure (Fig. 3), and multiple orders are generated by choosing different items as first item. The first item of the first sort order is selected as given in SortGreedy(); whereas, for remaining i sort orders, $1 < i \leq N$, the first item is randomly chosen from the initial 30 % items of first sort order.
- *InitializeGreedy* (q_j^t, GO) Here q_j^t is jth Q-bit individual in a population $Q(t)$. GO is a 2-D array containing 'n_s' sort orders. GO[i] refers to ith sort order. The procedure initializes the Q-bit individual q_j^t as explained in Sect. 4.1 using the sort order number GO[j mod n_s].
- *Make* $P(t)$ *from* $Q(t)$ The procedure collapses the Q-bits in $Q(t)$ to obtain solution in $P(t)$.
- *HamDistance* (p_j^s, b) Returns hamming distance between two binary strings p_j^s and b.
- *Mutate* (p_j^s) Mutates the solution p_j^s as described in Sect. 4.4.
- *Update* q_j^t based on b_j^t Rotates the Q-bits q_j^t towards bits in b_j^t as explained earlier and defined in using rotation angle as 0.01.

The maximum number of iterations in algorithm is controlled using a global constant, MaxIterations.

5 Results and discussion

The experiments are performed on a machine with Intel® Xeon® Processor E5645 (12 M Cache, 2.40 GHz, 5.86 GT/s Intel® QPI). The machine uses Red Hat Linux Enterprise 6. The program is written in C.

Almost all the problem instances considered here have been solved to optimality within 60 iterations, hence Max-

Fig. 7 Pseudo-code for QIEA-QKP

```
    Procedure QIEA-QKP
1   SolveGreedy(GSolution), MultipleSortGreedy(GreedyOrders, N);
2   t ← 0; b ← GSolution; we← GSolution; be← 0;
3   InitializeGreedy ( q_j^t, GreedyOrders) for each j ∈ {1, ... , n};
4   InitializeGreedy ( bqbit, GreedyOrders);
5   Make P(t) from Q(t);
6   RepairGreedy (P(t),GreedyOrders);
7   copy P(t) to B(t); we← min_{0≤i≤n}( b_i^t);be← max_{0≤i≤n}( b_i^t);
8   while ( t<MaxIterations) {
9     t ← t+1;
10    for r from 0 to η_1 do
11      for s from 0 to η_2 do
12          Make P(s) from Q(t);
13          RepairGreedy (P(s),GreedyOrders);
14          for each j ∈ {1, ... , n} if (HamDistance( p_j^s,b)<2) Mutate( p_j^s);
15          Evaluate P(s);
16          for each j ∈ {1, ... , n} if ( p_j^s better than p_j^t ) then p_j^t ← p_j^s;
17      } //*for s*//
18      for j ∈ {1, ... , n/2} ImproveLocal p_j^t ;for j ∈ {n/2, ... , n} RandImproveLocal p_j^t;
19      for each j ∈ {1, ... , n} if ( p_j^t is better than b_j^t) b_j^t← p_j^t;
20      for each j ∈ {1, ... , n} if (b_j^t is better than b) { b← b_j^t ; bqbit ← q_j^t;}
21      for each j ∈ {1, ... , n} Update q_j^t based on b_j^t ;
22      if(t mod PurgePeriod=0) StochasticPurge (B(t),be, we, Q(t),bqbit);
23    } //*for r*//
24    for each j ∈ {1, ... , n} Update q_j^t based on b;
25  } // *while*//
```

Iterations is set to 60. η_1 and η_2 and PurgePeriod are set empirically to 5 and population size is set to 160. Problems are named as n_d_i which specifies parameters size (n), density (d), seed (i) of an instance.

Two types of experiments are performed to observe the contribution of various modifications applied. First, the degree of degradation in performance due to removal of each modification from proposed QIEA-QKP is observed and studied. Second, the degree of improvement in performance due to introduction of each modification in SQIEA is observed and studied. Table 2 reports the performance observed in the first set of experiments. Table 3 reports the performance observed in the second set of experiments. The averages of the values taken over 40 BS benchmark instances of size 100, 40 instances of size 200 and 20 instances of size 300 are reported separately in these tables.

Table 2 shows the number of times the optimal solution is found in 30 runs, average time taken (in seconds) to reach optimal solution (T) and the average Function Evaluations (FEs) taken over 30 runs to reach best solution. Table 3 shows the number of times the optimal solution is found in 30 runs, average FEs taken to reach the best solution and relative percentage distance from optimal of their average value (RPD^{avg}) and relative percentage distance from optimal of their maximum value (RPD^{max}) obtained in 30 runs.

$$\text{RPD}^{\text{avg}} = \sum_{i=1}^{n} \text{RPD}_i^{\text{avg}}/n, \tag{3}$$

where n is the number of problems and $\text{RPD}_i^{\text{avg}}$ is relative percentage distance of average profit obtained in 30 runs from optimal value known for ith instance, such that

$$\text{RPD}_i^{\text{avg}} = (\text{OPT}_i - \text{AvgVal}_i) * 100/\text{OPT}_i, \tag{4}$$

where OPT_i and AvgVal_i are known optimal profit and average profit value obtained in 30 runs, respectively, for ith problem out of n problems considered.

Similarly,

$$\text{RPD}^{\text{max}} = \sum_{i=1}^{n} \text{RPD}_i^{\text{max}}/n, \tag{5}$$

where n is the number of problems and $\text{RPD}_i^{\text{avg}}$ is relative percentage distance of average profit obtained in 30 runs from optimal value known for ith instance, such that

$$\text{RPD}_i^{\text{max}} = (\text{OPT}_i - \text{MaxVal}_i) * 100/\text{OPT}_i, \tag{6}$$

where OPT_i and MaxVal_i are known optimal profit and maximum profit value obtained in 30 runs, respectively, for ith problem out of n problems considered.

In Table 4, for each feature proposed for improvement in QIEA-QKP, a study is shown in the improvement observed in SQIEA by including a feature in comparison with the degradation observed in QIEA-QKP on removing the feature.

The modifications are listed in order of their increasing effect on the performance of QIEA-QKP in Table 2. The following points are observed:

- StochasticPurge or mutation operators have very little effect on quality of solutions. However, they help in attaining good solutions in lesser number of FEs by systematically eliminating Q-bit individuals which perform poorly.
- Maximum degradation of performance is observed when sort orders used as domain knowledge in the algorithm are removed. The usage of sort orders in repair function and while initializing the Q-bit individuals enhances knowledge of areas containing good solutions during evolution which is useful.
- Stochastic Purging, Mutation or Initializing the best solution using the heuristic does not have much impact on the quality of solutions obtained but help in reducing the effort.
- If re-initialization of Q-bit individuals is not performed, computation effort is wasted on regions not having good solutions and there is a considerable degradation in quality of solutions.
- Randomized Improve results in better performance only for smaller problems ($n = 100$) while as size increases non-randomized Improve leads to better performance in terms of both the quality of solutions obtained and effort to reach the best solution.
- Applying randomized and non-randomized together in QIEA-QKP improves the performance more than when any one of them is applied individually.
- It is interesting to note that though local improve is computationally expensive it helps QIEA-QKP to reach the optimal solutions in fewer iterations and thus compensates for its overhead. Removing it not only reduces the quality of solutions obtained, but also increases the time taken to reach the best solution due to the resulting increase in the number of FEs.

The modifications are listed in order of their increasing effect on the performance of SQIEA in Table 3. Following points are observed.

- Inclusion of Mutation or StochasticPurge feature in SQIEA does not make substantial impact on quality of solutions or on effort required to reach best solution. However, removing any one of these features from the QIEA-QKP shows considerable degradation in performance. This implies that the Mutation and Sto-

Table 2 Effect of removing proposed modifications individually from QIEA-QKP

Ranking	Method	Size	100	200	300
I	QIEA-QKP	Hits	30	29.9	29.75
		T	0.02	0.56	1.65
		FEs	348.00	3369.11	4351.27
II	Without StochasticPurge	Hits	30	29.925	29.75
		T	0.04	0.75	2.05
		FEs	598.31	4236.23	5174.12
II	Without mutation	Hits	30	29.8	29.85
		T	0.03	0.77	2.09
		FEs	492.79	4477.79	5153.80
III	Without initializing best solution using heuristic	Hits	30	29.825	29.5
		T	0.06784	1.090925	3.611891
		FEs	750.8775	5606.271	10,375.83
IV	Without Re-initialization	Hits	29.95	29.875	29.45
		T	0.03	0.67	1.64
		FEs	482.47	3956.65	3947.18
V	Using improve without randomization	Hits	29.75	30	29.75
		T	0.11	0.89	2.33
		FEs	2198.86	3183.94	3431.29
VI	Using improve with randomization only	Hits	30	29.525	29
		T	0.04	1.15	5.05
		FEs	632.62	8658.56	19,753.28
VII	Without using improve	Hits	29.775	29.1	27.1
		T	0.08	0.85	3.60
		FEs	3698.59	11,218.29	25,385.38
VIII	Without using heuristic-based sort orders while initializing Q-bits (assigning $1/\sqrt{2}$ invariably to each Q-bit)	Hits	30	24.5	21.45
		T	0.28	4.33	13.02
		FEs	8168.40	34,519.55	51,893.11
IX	Without using heuristic-based sort orders while initializing the Q-bits and repairing the solutions	Hits	27.8	14.25	11.15
		T	1.22	2.10	6.18
		FEs	47,919.21	21,743.19	22,145.49

Hits number of runs reaching optimal solution, *T* time in seconds, *FEs* function evaluations

chasticPurge features contribute considerably only in combination with the other proposed modifications.

- Re-initialization of Q-bit individuals if added in even SQIEA shows a considerable improvement in the quality of solutions evident from increased number of function evaluations and decreased values of RPDs.
- Randomized Improve is more effective in smaller problems ($n = 100$) while non-randomized Improve is more effective in larger problems.
- Applying randomized and non-randomized Improve functions in combination provides advantages of both and improves performance of the algorithm more than when they are applied individually.
- Heuristic-based initialization of best solution when included in SQIEA, significantly improves its performance both in terms of quality of solutions and effort required to reach best. On the other hand, removing this feature from QIEA-QKP results in increased effort, but no degradation in quality of solutions. This is because the quality of solution is maintained due to other improvements in QIEA-QKP. This implies that improvements included in QIEA-QKP make it powerful enough to reach optimal solution even without initializing the solution using the heuristic.
- The maximum improvement in quality of solutions is observed when sort orders are used in repair function and Q-bit initialization.

A broad ranking of these features based on their contribution in improvement is clear from Tables 2, 3 and 4 as follows. Using heuristic-based sort orders while initializing the Q-bit individuals and repairing is first feature in the rank-

Table 3 Effect of including proposed improvements individually into SQIEA

Ranking	Method	Size	100	200	300
VIII	SQIEA	Hits	0.78	0.00	0.00
		RPD^{avg}	6.67428	17.50356	24.69355
		RPD^{max}	4.92968	15.83204	22.62981
		FEs	114,026.37	148,875.05	167,109.06
VIII	With mutation	Hits	0.75	0.00	0.00
		RPD^{avg}	6.72784	17.45152	24.70430
		RPD^{max}	4.75394	15.80447	22.98696
		FEs	114,306.68	149,767.75	166,326.95
VIII	With StochasticPurge	Hits	0.78	0.00	0.00
		RPD^{avg}	6.72870	17.47712	24.66387
		RPD^{max}	4.90731	15.86950	22.29675
		FEs	114,120.12	148,618.89	167,824.01
VII	With reinit	Hits	2.93	0.00	0.00
		RPD^{avg}	3.96410	15.27055	22.29491
		RPD^{max}	2.42350	13.54637	20.39589
		FEs	203,692.67	211,905.53	204,502.58
VI	With only randomized improve	Hits	7.10	1.98	0.00
		RPD^{avg}	2.83529	12.73331	20.35497
		RPD^{max}	1.78777	11.24463	18.11043
		FEs	133,938.93	198,444.25	223,725.06
V	With only non-randomized improve	Hits	5.73	1.05	0.00
		RPD^{avg}	2.02922	6.71729	7.77503
		RPD^{max}	1.22837	5.20773	6.38440
		FEs	73,399.47	110,089.42	126,848.55
IV	With both randomized and non-randomized improve on Q-bits individuals	Hits	7.98	2.03	0.00
		RPD^{avg}	1.56590	6.32361	7.70248
		RPD^{max}	0.65652	4.82354	6.12760
		FEs	109,396.47	153,909.39	176,335.22
III	With best solution initialized using heuristic	Hits	16.68	10.50	10.50
		RPD^{avg}	0.13298	0.07279	0.08452
		RPD^{max}	0.09258	0.06942	0.08452
		FEs	3851.97	1990.81	1.00
II	With sort orders used while repairing solutions (initializing Q-bits to $1/\sqrt{2}$)	Hits	28.20	19.30	13.90
		RPD^{avg}	0.00305	0.01752	0.13703
		RPD^{max}	0.00000	0.00230	0.05232
		FEs	41,578.27	89,689.40	113,272.44
I	With sort orders used while repairing solutions and initializing Q-bit individuals both	Hits	25.35	22.58	18.35
		RPD^{avg}	0.01226	0.00795	0.01398
		RPD^{max}	0.00000	0.00170	0.00038
		FEs	10,423.54	22,399.82	22,335.21

Hits number of runs reaching optimal solution, *FEs* function evaluations, RPD^{avg} relative percentage distance from optimal of their average value, RPD^{max} relative percentage distance from optimal of their maximum value

ing. The local improve heuristic is ranked second highest. Next in this ranking is the re-initialization of Q-bit individuals. The StochasticPurge and Mutation are next and last. The initialization of best solution using a heuristic is more effective when other heuristic-based modifications are not applied, otherwise its effect reduces.

Table 4 Contribution of each feature in improvement of QIEA-QKP over SQIEA

Sq. no.	Feature	Improvement in performance on the feature adding in SQIEA	Degradation in performance on removing the feature from QIEA-QKP
1	StochasticPurge	Little improvement in quality or effort	Very little effect on quality of solutions but considerable increase in number of FEs is observed
2	Mutation	Little improvement in quality or effort	Very little effect on quality of solutions but considerable increase in number of FEs is observed
3	Re-initialization (Reinit)	Considerable improvement in quality but number of FEs increases	Considerable degradation in quality of solutions is observed but little change in number of FEs
4	Initialization of best solution	Good improvement in quality of solutions with a tremendously decreased number of FEs	Very little effect on quality of solutions but considerable increase in number of FEs is observed
5	Local Improve	Considerable improvement in quality of solutions but little increase in FEs	Significant increase in FEs. Considerable degradation in quality
6	Randomized method of Improve	Considerable improvement in quality of solutions but marginal increase in FEs	Significant increase in FEs for small size (100) problems. Quality degraded for smaller problems but improved marginally for medium size problem
7	Non-randomized method of improve	Considerable improvement in quality of solutions but significant increase in Fes is not observed	Significant increase in FEs. Significant degradation in quality is not observed
8	Sort orders in initialization or repair	Significant improvement in quality of solutions, with considerable reduction in FEs is observed when sort orders are used in repair	Significant degradation in quality and effort on removing the use of sort orders in initialization
9	Sort orders in both repair and initialize	Maximum improvement in quality of solutions but considerable increase in FEs is observed	Maximum degradation in quality and effort

A more detailed analysis of the features and their contribution to the search process is given below with the discussion ordering the features in increasing order of determinism.

The features like Mutation, Stochastic Purge and Re-initialization increase the randomness. The latter two are applied on a need basis. Their removal from QIEA-QKP often results in an increased number of FEs for obtaining similar quality of solutions. This is because it may take time for the search to cover disparate regions of the search space possibly including the one with the best result found. Their incorporation in SQIEA does not show significant effect as it further increases the randomness of an already random search. These are effective only when they work in tandem with other deterministic techniques which favor exploitation of specific good regions or solutions.

Use of sort orders while initializing the Q-bits and/or repairing the observed solutions is most effective. This feature controls the randomness by biasing the mechanism of producing new solutions such that it produces solutions in specific good regions. However, it does not constrain the algorithm to restrict the search to a particular region in search space or towards a particular solution as it would only result in a local minimum. The addition of this feature in SQIEA improves the quality of solutions obtained and reduces time to obtain the best solution reduces dramatically. Conversely, removing it from QIEA-QKP the quality of solutions reduces and time to reach best solution increases significantly.

The improve function improves the solutions obtained by collapsing Q-bits and repairing the result. The improvement is typically a local search descent method using pairwise exchanges for all pairs. Random improve considers a subset of randomly selected pairs instead of all pairs. The other features help finding better regions and the improve functions perform hill-climbing for better solutions. Typically EAs are good at finding better regions of search space, but find it hard to pin point the optimal solutions. Therefore, this local search is a critical step. If it is removed, the number of FEs for reaching the best solutions is increased a lot.

The initialization of solution using a heuristic provides good initial points and regions for the search to start. Without this feature, the promising regions have to be found out by the QIEA itself. Since QIEA-QKP having balanced exploration and exploitation is good at this aspect even without this feature, this feature can be removed without causing much damage to the search process. Results are obtained marginally faster in QIEA-QKP if it is incorporated. However, in SQIEA this feature provides tremendous improvement.

On the basis of above discussion, an effective population-based search technique with right amount of randomness is developed, i.e., QIEA-QKP.

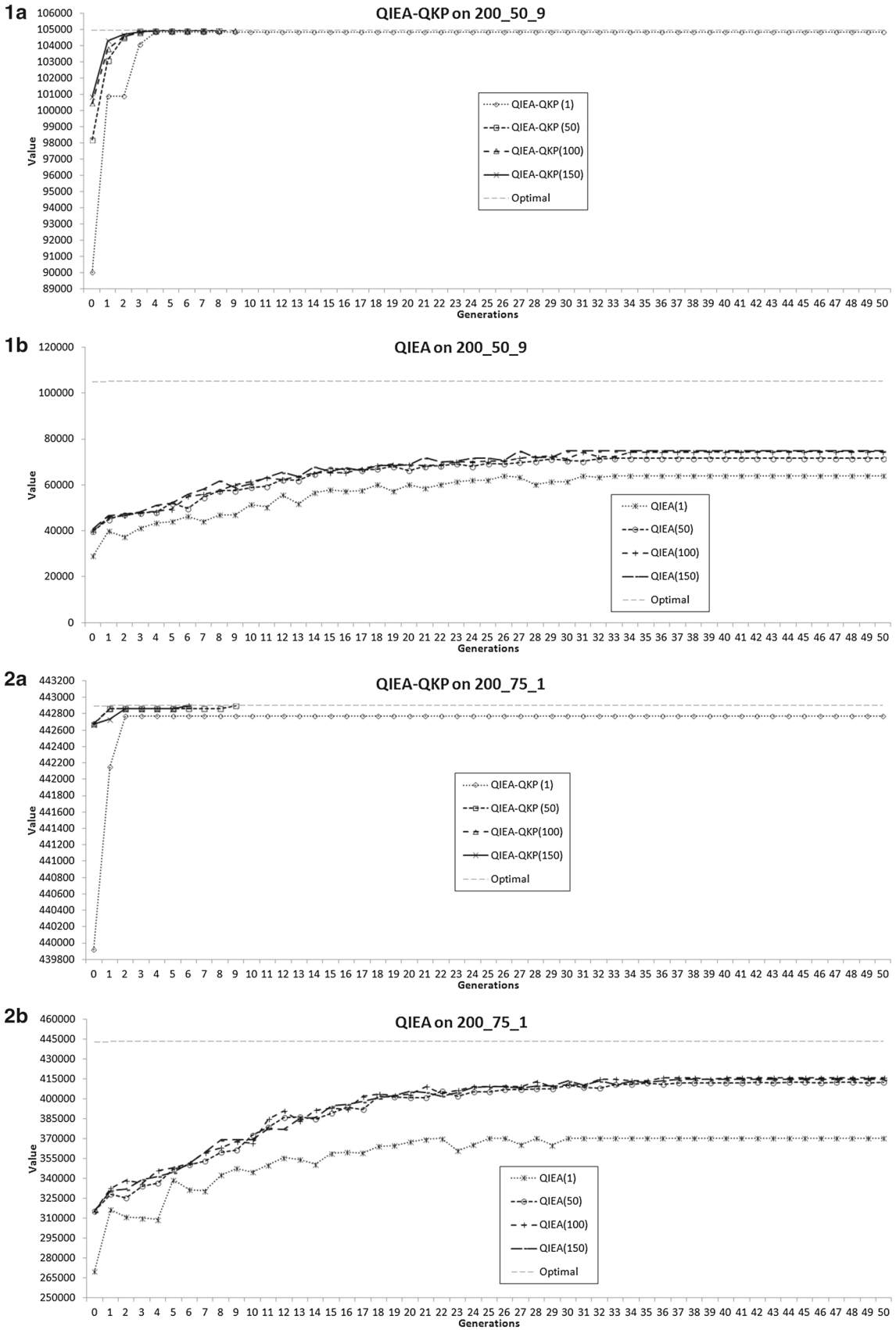

Fig. 8 Comparing convergence of SQIEA and QIEA-QKP with population sizes 1, 50, 100 and 150 for some selected problems. A plot is not extended further if it reaches optimal

Table 5 Comparison of SQIEA with proposed QIEA-QKP for BS benchmark instances of size 100

Problem	SQIEA						QIEA-QKP					
	Quality of value				Effort		Quality of value				Effort	
	Hits	Best	Avg	Stddev	AvgT	AvgFEs	Hits	Best	Avg	Stddev	AvgT	AvgFEs
100_25_1	0	17,177	16,219.2	289.36	1.87	121,252.20	**30**	**18,558**	**18,558**	**0.00**	**0.12**	**1957.43**
100_25_2	0	55,810	55,265.87	290.83	1.48	99,874.00	**30**	**56,525**	**56,525**	**0.00**	**0.00**	**1.00**
100_25_3	0	2755	2526.8	111.17	1.58	106,260.60	**30**	**3752**	**3752**	**0.00**	**0.02**	**89.30**
100_25_4	0	48,852	48,192.1	185.69	1.76	117,368.00	**30**	**50,382**	**50,382**	**0.00**	**0.19**	**7351.80**
100_25_5	0	61,205	61,152.63	45.06	1.03	69,516.20	**30**	**61,494**	**61,494**	**0.00**	**0.00**	**1.00**
100_25_6	0	35,898	35,638.33	115.59	1.85	120,767.60	**30**	**36,360**	**36,360**	**0.00**	**0.00**	**1.00**
100_25_7	0	12,895	12,498.57	231.90	2.00	130,831.20	**30**	**14,657**	**14,657**	**0.00**	**0.05**	**185.17**
100_25_8	0	19,809	19,394.27	180.00	1.93	125,581.60	**30**	**20,452**	**20,452**	**0.00**	**0.10**	**1166.77**
100_25_9	0	34,587	34,047.1	210.30	2.00	130,260.40	**30**	**35,438**	**35,438**	**0.00**	**0.02**	**131.33**
100_25_10	0	24,174	23,861.8	148.93	2.10	135,396.80	**30**	**24,930**	**24,930**	**0.00**	**0.00**	**1.00**
100_50_1	0	82,700	82,474.63	116.21	1.85	121,703.00	**30**	**83,742**	**83,742**	**0.00**	**0.02**	**79.10**
100_50_2	0	103,954	103,285.1	255.36	1.58	105,747.40	**30**	**104,856**	**104,856**	**0.00**	**0.02**	**89.20**
100_50_3	0	32,217	31,108.83	434.32	2.23	144,755.60	**30**	**34,006**	**34,006**	**0.00**	**0.10**	**1228.47**
100_50_4	0	103,927	103,670.1	178.12	1.57	105,636.00	**30**	**105,996**	**105,996**	**0.00**	**0.00**	**1.00**
100_50_5	0	55,443	54,950.57	272.24	2.13	137,420.00	**30**	**56,464**	**56,464**	**0.00**	**0.00**	**1.00**
100_50_6	0	13,525	12,642.23	367.92	1.89	124,763.40	**30**	**16,083**	**16,083**	**0.00**	**0.00**	**1.00**
100_50_7	0	51,502	50,779.43	255.75	2.15	138,126.40	**30**	**52,819**	**52,819**	**0.00**	**0.00**	**1.00**
100_50_8	0	53,604	53,086.83	384.66	2.08	133,405.80	**30**	**54,246**	**54,246**	**0.00**	**0.03**	**115.33**
100_50_9	0	67,549	67,239	198.70	2.04	132,976.40	**30**	**68,974**	**68,974**	**0.00**	**0.00**	**1.00**
100_50_10	0	87,349	86,440	404.79	1.86	123,356.80	**30**	**88,634**	**88,634**	**0.00**	**0.02**	**91.10**
100_75_1	**30**	**189,137**	**189,137**	**0.00**	0.01	710.00	**30**	**189,137**	**189,137**	**0.00**	**0.00**	**1.00**
100_75_2	0	94,410	93,482.27	547.71	2.14	138,439.00	**30**	**95,074**	**95,074**	**0.00**	**0.00**	**1.00**
100_75_3	0	59,407	57,408.8	698.92	2.22	143,410.80	**30**	**62,098**	**62,098**	**0.00**	**0.00**	**1.00**
100_75_4	0	71,605	70,252.13	708.58	2.11	135,650.60	**30**	**72,245**	**72,245**	**0.00**	**0.03**	**95.60**
100_75_5	0	22,630	21,887.27	441.50	1.96	129,020.40	**30**	**27,616**	**27,616**	**0.00**	**0.09**	**892.30**
100_75_6	0	142,424	142,178.2	84.80	1.47	98,133.60	**30**	**145,273**	**145,273**	**0.00**	**0.00**	**1.00**
100_75_7	0	109,488	109,106.4	237.11	1.92	125,753.80	**30**	**110,979**	**110,979**	**0.00**	**0.02**	**84.07**
100_75_8	0	16,350	15,409.7	417.83	1.88	124,391.60	**30**	**19,570**	**19,570**	**0.00**	**0.00**	**1.00**
100_75_9	0	101,823	101,476.7	160.25	1.89	123,310.00	**30**	**104,341**	**104,341**	**0.00**	**0.02**	**109.67**
100_75_10	0	141,687	141,173.4	313.04	1.42	94,473.00	**30**	**143,740**	**143,740**	**0.00**	**0.01**	**42.27**
100_100_1	0	79,185	77,636	932.96	2.10	135,856.00	**30**	**81,978**	**81,978**	**0.00**	**0.00**	**1.00**
100_100_2	0	189,352	18,7606	834.13	1.49	98,986.80	**30**	**190,424**	**190,424**	**0.00**	**0.02**	**106.60**
100_100_3	0	225,379	225,268.9	78.89	0.77	50,892.80	**30**	**225,434**	**225,434**	**0.00**	**0.00**	**1.00**
100_100_4	0	56,046	54,397.23	1008.50	1.86	122,028.80	**30**	**63,028**	**63,028**	**0.00**	**0.00**	**1.00**
100_100_5	0	229,677	229,258	114.04	0.96	64,352.00	**30**	**230,076**	**230,076**	**0.00**	**0.00**	**1.00**
100_100_6	0	71,359	68,970.43	960.84	2.24	144,770.60	**30**	**74,358**	**74,358**	**0.00**	**0.00**	**1.00**
100_100_7	0	7268	6624.3	316.57	1.51	101410.40	**30**	**10,330**	**10330**	**0.00**	**0.00**	**1.00**
100_100_8	0	59,520	57,501.13	1069.31	2.18	141,752.20	**30**	**62,582**	**62,582**	**0.00**	**0.00**	**1.00**
100_100_9	1	**232,754**	232,476.9	56.61	0.93	62,154.00	**30**	**232,754**	**232,754**	**0.00**	**0.00**	**1.00**
100_100_10	0	191,479	191,111.1	165.34	1.51	100,559.00	**30**	**193,262**	**193,262**	**0.00**	**0.02**	**82.53**
Avg	0.775			345.60	1.74	114,026.37	30			0.00	0.02	348.00

The best observed values for each parameter has been indicated in bold

Hits number of runs reaching optimal solution, *Avg* average profit, *Stddev* standard deviation in profit, *AvgT* average time in seconds, *AvgFEs* average number of FEs

Table 6 Comparison of SQIEA with proposed QIEA-QKP for BS benchmark instances of size 200.

Problem	SQIEA						QIEA-QKP					
	Quality of solutions				Effort		Quality of solutions				Effort	
	Hits	Best	Avg	Stddev	AvgT	AvgFEs	Hits	Best	Avg	Stddev	AvgT	AvgFEs
200_25_1	0	19,5258	193,582.1	672.11	6.79	136,484.00	**30**	**204,441**	**204,441**	**0.00**	**0.00**	**1.00**
200_25_2	0	236,842	236,193.9	228.12	5.06	103,102.20	**30**	**239,573**	**239,573**	**0.00**	**0.10**	**27.17**
200_25_3	0	245,076	244,577.8	166.20	3.64	74,715.40	**30**	**245,463**	**245,463**	**0.00**	**0.10**	**74.90**
200_25_4	0	217,158	216,252.8	251.22	5.64	115,373.00	**30**	**222,361**	**222,361**	**0.00**	**0.10**	**32.10**
200_25_5	0	180,093	178,825.8	536.09	6.75	144,680.60	**30**	**187,324**	**187,324**	**0.00**	**0.00**	**1.00**
200_25_6	0	66,268	64,637.8	734.26	8.13	171,737.60	**26**	**80,351**	**80,348.87**	**5.53**	**2.34**	**15699.81**
200_25_7	0	40,232	38,877.97	685.61	7.16	152,794.40	**30**	**59,036**	**59,036**	**0.00**	**0.83**	**1420.80**
200_25_8	0	141,217	139,434.8	814.02	8.52	180,769.40	**30**	**149,433**	**149,433**	**0.00**	**0.40**	**2182.60**
200_25_9	0	32,964	30,801.47	957.69	7.60	162,751.40	**30**	**49,366**	**49,366**	**0.00**	**0.01**	**1.00**
200_25_10	0	29,842	28,925.53	446.89	7.13	152,554.80	**30**	**48,459**	**48,459**	**0.00**	**0.01**	**1.00**
200_50_1	0	355,644	351,716.5	1416.23	7.05	149,537.80	**30**	**372,097**	**372,097**	**0.00**	**0.00**	**1.00**
200_50_2	0	194,062	191,088	1290.43	8.13	170,658.60	**30**	**211,130**	**211,130**	**0.00**	**2.49**	**19,513.27**
200_50_3	0	205,423	201,801.5	1647.44	8.55	179,772.00	**30**	**227,185**	**227,185**	**0.00**	**0.30**	**187.87**
200_50_4	0	208,969	205,833.3	1742.84	8.54	179,219.80	**30**	**228,572**	**228,572**	**0.00**	**0.01**	**1.00**
200_50_5	0	477,845	477,535.8	162.13	3.67	79,201.20	**30**	**479,651**	**479,651**	**0.00**	**0.00**	**1.00**
200_50_6	0	420,304	419,734.2	334.71	5.65	122,390.00	**30**	**426,777**	**426,777**	**0.00**	**0.11**	**86.97**
200_50_7	0	208,323	204,642.7	1414.43	8.46	176,806.80	**30**	**220,890**	**220,890**	**0.00**	**0.92**	**5275.43**
200_50_8	0	302,123	299,882.1	1147.11	8.12	169,333.00	**30**	**317,952**	**317,952**	**0.00**	**0.30**	**810.43**
200_50_9	0	76,337	74,055.43	1151.30	7.48	157,455.00	**30**	**104,936**	**104,936**	**0.00**	**2.19**	**12,681.27**
200_50_10	0	271,854	268,834.3	1483.16	7.82	162,835.60	**30**	**284,751**	**284,751**	**0.00**	**0.30**	**596.50**
200_75_1	0	421,113	416,091.5	1683.47	7.72	162,802.20	**30**	**442,894**	**442,894**	**0.00**	**2.09**	**20,518.40**
200_75_2	0	251,155	246,865.2	2188.81	8.73	180,871.00	**30**	**286,643**	**286,643**	**0.00**	**5.54**	**46,580.30**
200_75_3	0	27,918	25,453.07	931.90	6.60	138,581.00	**30**	**61,924**	**61,924**	**0.00**	**0.40**	**526.17**
200_75_4	0	74,617	72,434.97	1035.95	7.80	162,284.60	**30**	**128,351**	**128,351**	**0.00**	**0.01**	**1.00**
200_75_5	0	80,841	78,021.17	1620.23	6.94	147,109.80	**30**	**137,885**	**137,885**	**0.00**	**0.01**	**1.00**
200_75_6	0	183,975	179,061.8	2229.60	8.17	171,629.00	**30**	**229,631**	**229,631**	**0.00**	**0.87**	**1544.67**
200_75_7	0	217,881	210,524.4	2980.01	8.24	173,108.20	**30**	**269,887**	**269,887**	**0.00**	**0.66**	**1153.97**
200_75_8	0	588,567	586,445.8	1808.62	6.54	138,146.20	**30**	**600,858**	**600,858**	**0.00**	**0.00**	**1.00**
200_75_9	0	495,443	493,553.5	1245.91	7.20	152,808.60	**30**	**516,771**	**516,771**	**0.00**	**0.13**	**93.93**
200_75_10	0	94,451	89,752.5	1683.20	7.29	155,314.20	**30**	**142,694**	**142,694**	**0.00**	**0.01**	**1.00**
200_100_1	0	936,095	935,171.7	426.01	4.11	89,178.00	**30**	**937,149**	**937,149**	**0.00**	**0.11**	**84.90**
200_100_2	0	232,202	221,884.4	3463.59	7.86	166,320.20	**30**	**303,058**	**303,058**	**0.00**	**0.84**	**1006.97**
200_100_3	0	15,337	13,602.3	609.60	5.77	124,916.40	**30**	**29,367**	**29,367**	**0.00**	**0.14**	**89.77**
200_100_4	0	47,694	45,347.73	1100.51	6.63	142,526.60	**30**	**100,838**	**100,838**	**0.00**	**0.01**	**1.00**
200_100_5	0	767,658	764,253.7	3247.37	6.44	137,820.40	**30**	**786,635**	**786,635**	**0.00**	**0.12**	**88.40**
200_100_6	0	27,152	24,487.3	922.73	6.05	129,999.40	**30**	**41,171**	**41,171**	**0.00**	**0.01**	**1.00**
200_100_7	0	673,896	671,673.8	2566.76	7.37	156,884.20	**30**	**701,094**	**701,094**	**0.00**	**0.12**	**84.23**
200_100_8	0	759,626	753,016.3	1959.81	7.04	151,425.00	**30**	**782,443**	**782,443**	**0.00**	**0.47**	**4300.07**
200_100_9	0	598,531	594,475.8	2826.73	7.29	154,267.80	**30**	**628,992**	**628,992**	**0.00**	**0.13**	**89.63**
200_100_10	0	328,817	322,073.3	3641.52	8.44	176,836.60	**30**	**378,442**	**378,442**	**0.00**	**0.01**	**1.00**
Avg	0			1386.36	7.05	148,875.05	29.9			0.14	0.56	3369.11

The best observed values for each parameter has been indicated in bold

Hits number of runs reaching optimal solution, *Avg* average profit, *Stddev* standard deviation in profit, *AvgT* average time in seconds, *AvgFEs* average number of FEs

Table 7 Comparison of SQIEA with proposed QIEA-QKP for BS benchmark instances of size 300

Problem	SQIEA						QIEA-QKP					
	Quality of solution				Effort		Quality of solution				Effort	
	Hits	Best	Avg	Stddev	AvgT	AvgFEs	Hits	Best	Avg	Stddev	AvgT	AvgFEs
300_25_1	0	9793	8652.667	422.25	13.68	134,758.60	**30**	**29,140**	**29,140**	**0.00**	**0.05**	**1.00**
300_25_2	0	249,595	245,634.7	1630.10	20.12	192,098.60	**29**	**281,990**	**281,989.4**	**3.10**	**3.36**	**9347.45**
300_25_3	0	185,605	181,958.5	1586.75	21.07	196,778.80	**30**	**231,075**	**231,075**	**0.00**	**0.03**	**1.00**
300_25_4	0	428,258	425,099.4	1003.31	16.65	160,597.20	**30**	**444,759**	**444,759**	**0.00**	**0.98**	**2886.63**
300_25_5	0	6855	5743.167	268.51	12.97	124,436.80	**30**	**14,988**	**14,988**	**0.00**	**3.38**	**151,11.43**
300_25_6	0	230,876	226,724.6	2181.99	20.91	200,165.20	**30**	**269,782**	**269,782**	**0.00**	**1.72**	**2058.60**
300_25_7	0	471,931	470,657.1	609.53	15.63	153,466.40	**30**	**485,263**	**485,263**	**0.00**	**0.51**	**252.43**
300_25_8	0	3387	2999.1	137.73	11.21	110,695.00	**30**	**9343**	**9343**	**0.00**	**1.06**	**2349.20**
300_25_9	0	214,708	207,187.3	2399.78	20.48	197,481.20	**26**	**250,761**	**250,759.7**	**3.46**	**11.63**	**42,638.73**
300_25_10	0	360,844	356,019.2	1492.99	17.90	173,352.80	**30**	**383,377**	**383,377**	**0.00**	**0.01**	**1.00**
300_50_1	0	448,176	442,527.3	3107.13	20.06	193,855.40	**30**	**513,379**	**513,379**	**0.00**	**1.92**	**2319.40**
300_50_2	0	48,000	45,135.63	1092.04	15.12	148,840.40	**30**	**105,543**	**105,543**	**0.00**	**0.05**	**1.00**
300_50_3	0	839,404	834,913.8	1832.46	17.24	168,376.40	**30**	**875,788**	**875,788**	**0.00**	**0.53**	**224.63**
300_50_4	0	196,597	188,723.9	2560.05	18.53	178,334.00	**30**	**307,124**	**307,124**	**0.00**	**4.90**	**6533.60**
300_50_5	0	663,770	658,652.2	2883.20	19.46	188,890.80	**30**	**727,820**	**727,820**	**0.00**	**0.96**	**1390.97**
300_50_6	0	685,417	681,156.8	2263.49	18.75	183,437.00	**30**	**734,053**	**734,053**	**0.00**	**0.01**	**1.00**
300_50_7	0	17,934	15,420	850.46	13.51	132,586.00	**30**	**43,595**	**43,595**	**0.00**	**1.23**	**1758.07**
300_50_8	0	726,558	719,336.4	2761.22	18.49	179,859.20	**30**	**767,977**	**767,977**	**0.00**	**0.59**	**147.17**
300_50_9	0	709,592	703,244.5	3279.21	18.91	183,854.80	**30**	**761,351**	**761,351**	**0.00**	**0.01**	**1.00**
300_50_10	0	975,666	972,426.6	2186.64	14.31	140,316.60	**30**	**996,070**	**996,070**	**0.00**	**0.01**	**1.00**
Avg	0			1727.44	17.25	167,109.06	29.75			0.33	1.65	4351.27

The best observed values for each parameter has been indicated in bold

Hits number of runs reaching optimal solution, *Avg* average profit, *Stddev* standard deviation in profit, *AvgT* average time in seconds, *AvgFEs* average number of FEs

The trend of convergence in SQIEA and QIEA-QKP is studied with different population sizes, viz. 1, 50, 100 and 150 through 50 generations by plotting the best profit values obtained after each generation. The plots (drawn till they reach the optimal first time) are shown for selected problems 200_50_9 and 200_75_1 in Fig. 8(1a, b) and (2a, b), respectively. The optimal value is shown using a gray dotted line. It is quite clear that QIEA-QKP converges to near optimal value which is far better than value SQIEA converges to. Moreover, QIEA-QKP obtains it in substantially less number of FEs. The increment in population size further improves the performance of algorithm on both aspects, i.e., improved quality in less number of generations. Population size of 150 is sufficient for these problems. Tables 5, 6 and 7 present the detailed comparison of SQIEA with proposed QIEA-QKP on all the 100 BS benchmark instances. Following results have been shown in these tables, number of runs reaching optimal solution (Hits), best, average (AVG) and standard deviation in values obtained (Stddev), average time in seconds (AvgT) and average number of FEs (AvgFEs). It is clear that QIEA-QKP is markedly better than the SQIEA.

Table 5 shows that QIEA-QKP returns optimal solution for all problems of size 100 variables in 30 runs taking 0.02 s on an average and requiring around 348 FEs per trial on an average for all 40 instances.

Table 6 shows that QIEA-QKP provides optimal solution for problems of size 200 variables in 99.67 % 30 runs in less than 0.6 s requiring around 3369 FEs per trial on an average for all 40 instances.

Table 7 shows that QIEA-QKP provides optimal solution for problems of size 300 variables in 99.17 % of 30 runs in less than 1.7 s requiring around 4351 FEs per trial on an average for all 20 instances.

Table 8 shows the comparison of proposed QIEA-QKP with known best QIEA for QKP, i.e., QIEA-PPA on 20 BS benchmark instances. Since QIEA-PPA performs better only in terms of quality of solutions while it requires more FEs as compared to GGA, results of GGA are also included while comparing QIEA-QKP with QIEA-PPA. Table 8 presents number of times optimal solution is reached in 50 runs and

Table 8 Comparison of QIEA-QKP with GGA and QIEA-PPA

Problem	Opt	Julstroms greedy EA			QIEA-PPA			QIEA-QKP		
		Hits	BestFEs	AvgFEs	Hits	BestFEs	AvgFEs	Hits	BestFEs	AvgFEs
100_25_1	18,558	**50**	375	**2269**	**50**	4250	182,716	**50**	**20**	3177.94
100_25_2	56,525	**50**	75	187	**50**	50	50	**50**	**1**	**1**
100_25_3	3752	36	625	9203	46	4850	289,376	**50**	**82**	**90.62**
100_25_4	50,382	23	175	**4849**	**50**	50	50	**50**	**5**	5860.12
100_25_5	61,494	**50**	25	54	**50**	50	50	**50**	**1**	**1**
100_25_6	36,360	**50**	375	1330	**50**	50	50	**50**	**1**	**1**
100_25_7	14,657	**50**	275	486	**50**	50	33,228	**50**	**22**	**194.88**
100_25_8	20,452	**50**	275	**411**	**50**	50	50	**50**	**81**	1095.44
100_25_9	35,438	37	225	11,777.5	**50**	150	25,434	**50**	**35**	**118.26**
100_25_10	24,930	**50**	325	582	38	6050	192,471	**50**	**1**	**1**
200_100_1	937,149	**50**	1550	46,990	**50**	1100	54,816	**50**	**82**	**84.88**
200_100_2	303,058	**50**	2950	10,362	49	1500	207,634	**50**	**148**	**1010.22**
200_100_3	29,367	**50**	1250	1940	**50**	3700	153,232	**50**	**25**	**97.74**
200_100_4	100,838	**50**	1250	2082	**50**	100	100	**50**	**1**	**1**
200_100_5	786,635	**50**	1650	4928	41	1900	267,374	**50**	**82**	**89.18**
200_100_6	41,171	**50**	1150	1392	**50**	100	100	**50**	**1**	**1**
200_100_7	701,094	**50**	1850	19,666	40	700	90,540	**50**	**82**	**84.24**
200_100_8	782,443	6	107,950	157,100	41	5100	567,602	**49**	**82**	**5613.1224**
200_100_9	628,992	**50**	1550	5716	42	2500	523,990	**50**	**82**	**91.08**
200_100_10	378,442	**50**	3950	17,932	47	7700	221,086	**50**	**1**	**1**
Sum		902			944			999		

The best observed values for each parameter has been indicated in bold
Opt optimal value of Profit, *Hits* number of runs reaching optimal solution, *BestFEs* average number of FEs, *AvgFEs* average number of FEs

minimum (BestFEs) and average (AvgFEs) function evaluations required within 50 runs to solve each of the 20 instances. GGA provides optimal solution 902 times, QIEA-PPA provides optimal solution 944 times and outperforming both QIEA-QKP provides optimal solution 999 times out of 1000 (50 runs for each of 20 instances). GGA requires 14,962.83 FEs on an average, QIEA-PPA requires 140,497.5 FEs on an average and again outperforming both QIEA-QKP requires only 880.73 FEs on an average over total 1000 trials. It is clear that QIEA-QKP outperforms both QIEA-PPA and GGA in terms of frequency of reaching optimal and computational effort required.

Table 9 shows the comparison of proposed QIEA-QKP with the a popular population-based Mini-Swarm algorithm given by Xie and Liu (2007) using BS benchmark instances of size 100 and 200 variables. Table 9 presents number of times optimal solution is reached in 100 runs (Hits), average value over 100 runs of relative percentage deviation (RPD) from the optimal and average time taken (AvgT) in seconds required to reach best solution. Mini-Swarm algorithm provides optimal solution 96 times on average out of 100 runs for problems of size 100 and 93 times for size 200. On the other hand QIEA-QKP provides 100 times for problems of size 100 and 99.825 times on an average out of 100 runs for problems of size 200. QIEA-QKP takes considerably less time than Mini-Swarm algorithm for the purpose. So QIEA-QKP clearly outperforms the Mini-Swarm algorithm.

6 Conclusions

This work presents a QIEA which is enhanced using several features to improve its capability to exploit and explore the solution space. The detailed comparison and analysis of effects of all the features applied is presented. Various versions of QIEA-QKP having different combinations of features are used for analysis. A ranking is done using a systematic method based on the performance of various versions of QIEA-QKP. The analysis is further used to rank the modifications based on their individual contribution to the improvements in QIEA-QKP.

QKP is a difficult NP-hard optimization problem as no pseudo-polynomial time algorithm exists to solve it. The proposed QIEA-QKP is shown to provide optimal solutions in

Table 9 Comparison of QIEA-QKP with Mini-Swarm algorithm

Problem	Mini-swarm			QIEA-QKP			Problem	Mini-Swarm			QIEA-QKP		
	Hits	RPD	AvgT	Hits	RPD	AvgT		Hits	RPD	AvgT	Hits	RPD	AvgT
100_25_1	**100**	0	0.430	**100**	**0**	**0.180**	**200_25_1**	**100**	0	1.221	**100**	**0**	**0.003**
100_25_2	**100**	0	0.170	**100**	**0**	**0.001**	**200_25_2**	**100**	0	0.499	**100**	**0**	**0.095**
100_25_3	**100**	0	0.190	**100**	**0**	**0.020**	**200_25_3**	98	7E-04	0.536	**100**	**0**	**0.105**
100_25_4	**100**	0	0.920	**100**	**0**	**0.130**	**200_25_4**	**100**	0	0.726	**100**	**0**	**0.098**
100_25_5	**100**	0	0.064	**100**	**0**	**0.001**	**200_25_5**	**100**	0	1.571	**100**	**0**	**0.003**
100_25_6	**100**	0	0.415	**100**	**0**	**0.001**	**200_25_6**	28	0.054	2.781	**93**	**0.001**	2.680
100_25_7	**100**	0	0.233	**100**	**0**	**0.051**	**200_25_7**	77	0.036	2.583	**100**	**0**	**0.834**
100_25_8	**100**	0	0.313	**100**	**0**	**0.096**	**200_25_8**	**100**	0	2.038	**100**	**0**	**0.417**
100_25_9	39	0.117	0.801	**100**	**0**	**0.021**	**200_25_9**	**100**	0	1.405	**100**	**0**	**0.012**
100_25_10	**100**	0	0.890	**100**	**0**	**0.002**	**200_25_10**	**100**	0	0.940	**100**	**0**	**0.013**
100_50_1	**100**	0	0.273	**100**	**0**	**0.017**	**200_50_1**	**100**	0	1.292	**100**	**0**	**0.003**
100_50_2	54	0.028	0.224	**100**	**0**	**0.017**	**200_50_2**	26	0.005	1.825	**100**	**0**	2.384
100_50_3	88	0.008	0.366	**100**	**0**	**0.114**	**200_50_3**	**100**	0	1.606	**100**	**0**	**0.284**
100_50_4	**100**	0	0.137	**100**	**0**	**0.001**	**200_50_4**	**100**	0	4.801	**100**	**0**	**0.007**
100_50_5	**100**	0	0.669	**100**	**0**	**0.001**	**200_50_5**	**100**	0	0.501	**100**	**0**	**0.001**
100_50_6	**100**	0	0.263	**100**	**0**	**0.002**	**200_50_6**	**100**	0	1.053	**100**	**0**	**0.113**
100_50_7	**100**	0	0.417	**100**	**0**	**0.001**	**200_50_7**	**100**	0	2.066	**100**	**0**	**0.903**
100_50_8	**100**	0	0.416	**100**	**0**	**0.032**	**200_50_8**	**100**	0	1.726	**100**	**0**	**0.290**
100_50_9	**100**	0	0.950	**100**	**0**	**0.001**	**200_50_9**	**100**	0	**1.097**	**100**	**0**	2.376
100_50_10	**100**	0	0.340	**100**	**0**	**0.019**	**200_50_10**	98	4E-05	2.079	**100**	**0**	**0.294**
100_75_1	**100**	0	0.090	**100**	**0**	**0.001**	**200_75_1**	16	0.025	3.339	**100**	**0**	**2.286**
100_75_2	**75**	0.014	0.749	**100**	**0**	**0.001**	**200_75_2**	95	5E-05	**2.797**	**100**	**0**	5.705
100_75_3	**100**	0	0.349	**100**	**0**	**0.002**	**200_75_3**	**100**	0	0.656	**100**	**0**	**0.415**
100_75_4	**100**	0	0.522	**100**	**0**	**0.025**	**200_75_4**	**100**	0	1.232	**100**	**0**	**0.012**
100_75_5	**100**	0	0.210	**100**	**0**	**0.087**	**200_75_5**	**100**	0	1.127	**100**	**0**	**0.013**
100_75_6	**100**	0	0.555	**100**	**0**	**0.001**	**200_75_6**	**100**	0	1.661	**100**	**0**	**0.872**
100_75_7	**100**	0	0.341	**100**	**0**	**0.019**	**200_75_7**	**100**	0	6.291	**100**	**0**	**0.644**
100_75_8	**100**	0	0.192	**100**	**0**	**0.002**	**200_75_8**	98	2E-04	1.966	**100**	**0**	**0.002**
100_75_9	**100**	0	0.417	**100**	**0**	**0.026**	**200_75_9**	**100**	0	1.515	**100**	**0**	**0.131**
100_75_10	**100**	0	0.209	**100**	**0**	**0.014**	**200_75_10**	**100**	0	1.072	**100**	**0**	**0.013**
100_100_1	**100**	0	0.271	**100**	**0**	**0.002**	**200_100_1**	**100**	0	0.574	**100**	**0**	**0.111**
100_100_2	**100**	0	0.230	**100**	**0**	**0.019**	**200_100_2**	**100**	0	1.179	**100**	**0**	**0.850**
100_100_3	**100**	0	0.190	**100**	**0**	**0.001**	**200_100_3**	**100**	0	0.301	**100**	**0**	**0.148**
100_100_4	**100**	0	0.400	**100**	**0**	**0.002**	**200_100_4**	**100**	0	0.543	**100**	**0**	**0.014**
100_100_5	**100**	0	0.106	**100**	**0**	**0.001**	**200_100_5**	**100**	0	2.043	**100**	**0**	**0.121**
100_100_6	**100**	0	0.330	**100**	**0**	**0.002**	**200_100_6**	**100**	0	0.311	**100**	**0**	**0.014**
100_100_7	**100**	0	0.120	**100**	**0**	**0.002**	**200_100_7**	**100**	0	1.732	**100**	**0**	**0.119**
100_100_8	**100**	0	0.257	**100**	**0**	**0.002**	**200_100_8**	**100**	0	1.258	**100**	**0**	**0.677**
100_100_9	**100**	0	0.109	**100**	**0**	**0.001**	**200_100_9**	**100**	0	2.277	**100**	**0**	**0.135**
100_100_10	**100**	0	0.240	**100**	**0**	**0.015**	**200_100_10**	**100**	0	1.757	**100**	**0**	**0.009**
Average	**96**	**0.004**	**0.359**	**100**	**0**	**0.023**		**93**	**0.003**	**1.649**	**99.825**	**3E-05**	**0.58**

The best observed values for each parameter has been indicated in bold

more than 99.5 % runs for benchmark instances of size up to 300 variables within around 1 s per trial per instance on an average.

A comparison of QIEA-QKP with existing SQIEA shows that proposed QIEA-QKP is substantially better both in terms of quality of solutions obtained and computational

effort required for QKP. Comparisons show that the proposed QIEA-QKP outperforms some existing population-based algorithms like a Genetic Algorithm, an existing QIEA and a popular Mini-Swarm algorithm.

This research achieves a modified QIEA framework where known effective heuristics for QKP and well-known operators of EAs like mutation, re-initialization and purging are used in combination to improve the random search. The effectiveness of this approach is evident from its significantly improved performance on well-known benchmark instances. A systematic way of studying the importance of each modification as well as ranking them is presented which can be used to apply them appropriately.

The proposed QIEA-QKP is a population-based search, in which several features are incorporated. Some of these features increase the randomness in the search process for better exploration and the others compensate by local search for better exploitation together enabling a judicious combination tailored for particular problem being solved, hence referred as "right-sizing the randomness" in the QIEA search.

The improved QIEA framework presented can be used for other similar NP-hard problems too.

Acknowledgements The authors are grateful to Department of Science and Technology, India (DST) and Deutsche Forschungsgemeinschaft, Germany (DFG) for the support under Project No. INT/FRG/DFG/P-38/2012 titled "Algorithm Engineering of Quantum Evolutionary Algorithms for Hard Optimization Problems".

Compliance with ethical standards

Conflicts of interest None.

References

Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford

Billionet A, Soutif E (2004) QKP instances. [Online] http://cedric.cnam.fr/soutif/QKP/QKP.html. Accessed 15 February 2013

Caprara A, Pisinger D, Toth P (1999) Exact solution of the quadratic knapsack problem. Inf J Comput 11(2):125–137

Dijkhuijen G, Faigle U (1993) A cutting-plane approach to he edge-weighted maximal clique problem. Eur J Oper Res 69:121–130

Engebretsen L, Holmerin J (2000) Clique is hard to approximate within n1-o(1). In: Proceedings of 27th international colloquium, ICALP 2000, vol 1853. Springer, Berlin, pp 2–12

Ferreira CE, Martin A, deSouza CC (1996) Formulations and valid inequalities for node capacitated graph partitioning. Math Program 74:247–266

Gallo G, Hammer P, Simeone B (1980) Quadratic knapsack problems. Math Program Study 12:132–149

Han K-H (2006) On the analysis of the quantum-inspired evolutionary algorithm with a single individual. In: Proceedings of CEC'06, Vancouver, Canada, pp 2622–2629

Han K-H, Kim J-H (2002) Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans Evolut Comput 6(6):580–593

Han K-H, Kim J-H (2003) On setting the parameters of quantum-inspired evolutionary algorithm for practical application. In: Proceedings of CEC 2003, vol 1, pp 178–194

Han K-H, Kim J-H (2004) Quantum-inspired evolutionary algorithms with a new termination criterion, h-epsilon gate, and two-phase scheme. IEEE Trans Evolut Comput 8(2):156–169

Han K, Park K, Lee C, Kim J (2001) Parallel quantum-inspired genetic algorithm for combinatorial optimization problem. In: Proceedings of CEC 2001, vol 2, pp 1422–1429

Imabeppu T, Nakayama S, Ono S (2008) A study on a quantum-inspired evolutionary algorithm based on pair swap. Artif Life Robot 12:148–152

Johnson E, Mehrotra A, Nemhauser G (1993) Min-cut clustering. Math Program 62:133–151

Julstorm BA (2005) Greedy, genetic and greedy genetic algorithms for the quadratic knapsack problem. In: Proceedings of GECCO 2005, Washingtom DC, USA, pp 607–614

Kellerer H, Pferschy U, Pisinger D (2004) Knapsack problems. Springer, Berlin

Kim Y, Kim JH, Han KH (2006) Quantum-inspired multiobjective evolutionary algorithm for multiobjective 0/1 knapsack problems. In: Proceedings of CEC 2006, pp 2601–2606

Laghhunn DL (1970) Quadratic binary programming with applications to capital budgetting problems. Oper Res 18:454–461

Li Z, Rudolph G, Li K (2009) Convergence performance comparison of quantum-inspired multi-objective evolutionary algorithms. Comput Math Appl 57:1843–1854

Lu TC, Yu GR (2013) An adaptive population multi-objective quantum-inspired evolutionary algorithm for multi-objective 0/1 knapsack problems. Inf Sci 243:39–56

Mahdabi P, Jalili S, Abadi M (2008) A multi-start quantum-inspired evolutionary algorithm for solving combinatorial optimization problems. In: Proceedings of GECCO '08, pp 613–614

Nowotniak R, Kucharski J (2012) GPU-based tuning of quantum-inspired genetic algorithm for a combinatorial optimization problem. Bull Pol Acad Sci Tech Sci 60(2):323–330

Park K, Lee K, Park S (1996) An extended formulation approach to the edge weighted maximal cliqur problem. Eur J Oper Res 95:671–682

Patvardhan C, Narayan A, Srivastav A (2007) Enhanced quantum evolutionary algorithms for difficult knapsack problems. In: Proceedings of PReMI'07. Springer, Berlin, pp 252–260

Patvardhan C, Prakash P, Srivastav A (2012) A novel quantum-inspired evolutionary algorithm for the quadratic knapsack problem. Int J Math Oper Res 4(2):114–127

Patvardhan C, Bansal S, Srivastav A (2014a) Solution of "Hard" knapsack instances using quantum inspired evolutionary algorithm. Int J Appl Evolut Comput 5(1):52–68

Patvardhan C, Bansal S, Srivastav A (2014b) Balanced quantum-inspired evolutionary algorithm for multiple knapsack problem. Int J Intell Syst Appl 11:1–11

Pisinger D (2007) The quadratic knapsack problem—a survey. Discrete Appl Math 155:623–648

Platel M D, Schliebs S, Kasabov N (2007) A versatile quantum-inspired evolutionary algorithm. In: Proceedings of CEC'07, pp 423–430

Platel MD, Schliebs S, Kasabov N (2009) Quantum-inspired evolutionary algorithm: a multimodel EDA. IEEE Trans Evolut Comput 13(6):1218–1232

Qin Y, Zhang G, Li Y, Zhang H (2012) A comprehensive learning quantum-inspired evolutionary algorithm. In: Qu X, Yang Y (eds) LNCS, IBI 2011, Part-II, CCIS 268, pp 151–157

Rhys J (1970) A selection problem of shared fixed costs and network flows. Manag Sci 17:200–207

Tayarani-N M-H, Akbarzadeh-T M-R (2008) A sinusoid size ring structure quantum evolutionary algorithm. In: Proceedings of IEEE conference on cybernetics and intelligent systems, pp 1165–1170

Witzall C (1975) Mathematical methods of site selection for electronic message system (EMS). In: Technical report, NBS internal report Washington, DC. Operational Research Section

Xie X, Liu J (2007) A mini-swarm for the quadratic knapsack problem. In: Proceedings of IEEE swarm intelligence symposium, Honolulu, USA, pp 190–197

Zhang G, Gheorghe M, Wu CZ (2008) A quantum-inspired evolutionary algorithm based on p systems for knapsack problem. Fund Inf 87(1):93–116

Zhang G (2011) Quantum-inspired evolutionary algorithms: a survey and empirical study. J Heurist 17(3):303–351

Zhang G, Gheorghe M, Li Y (2012) A membrane algorithm with quantum-inspired subalgorithms and its application to image processing. Nat Comput 11:701–707

Zhao Z, Peng X, Peng Y, Yu, E (2006) An effective repair procedure based on quantum-inspired evolutionary algorithm for 0/1 knapsack problems. In: Proceedings of the 5th WSEAS int. conf. on instrumentation, measurement, circuits and systems, Hangzho, pp 203–206