Memetic Comp. (2015) 7:135–155
DOI 10.1007/s12293-015-0162-1

REGULAR RESEARCH PAPER

Quantum-Inspired Evolutionary Algorithm for difficult knapsack problems

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

Received: 24 August 2013 / Accepted: 6 April 2015 / Published online: 21 April 2015

Abstract Quantum Inspired Evolutionary Algorithms (QIEAs) are Evolutionary Algorithms which use concepts and principles of quantum computing. The 0/1 knapsack problem (KP) is a well known combinatorial optimization problem that has been typically used to validate the performance of QIEAs. However, there are some variants of KPs called difficult knapsack problems (DKPs) that are known to be more difficult to solve. QIEAs have not yet been fully explored for solving these. In this work, an improved QIEA, called QIEA-PSA is presented. A novel method to initialize the qubit individuals based on heuristic information for the KP instance and a method for size reduction for each new generation are introduced in the presented QIEA-PSA. Experiments are carried out for several types of DKPs that are much larger in size than those attempted hitherto. QIEA-PSA provides much better solutions than QIEA with much lesser computation times. Even a serial implementation of QIEA-PSA competes favorably on the same problem instances with a parallel implementation of an exact algorithm given recently in literature. A comparison is made which shows QIEA-PSA outperforms a recently applied population based search technique to solve benchmark KP instances. The ideas used for developing QIEA-PSA are general and may be utilized with advantage on other problems.

Keywords Quantum Inspired Evolutionary Algorithm · Combinatorial optimization · Knapsack problem

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

C. Patvardhan
cpatvardhan@googlemail.com

Anand Srivastav
asr@numerik.uni-kiel.de

[1] Department of Electrical Engineering, Faculty of Engineering, DEI, Agra 282005, India

[2] Christian-Albrechts-Universität zu Kiel, Institut für Informatik, Christian-Albrechts-Platz 4, 24118 Kiel, Germany

[3] 1/25, Hazuri Bhawan, Peepal Mandi, Agra 282003, India

1 Introduction

The knapsack problem in its most basic form is defined as follows: Given a set of n items with their profits p_j and weights w_j, the problem is to select a subset of items such that profit is maximized and weight does not exceed the capacity C.

$$\begin{aligned} &\text{maximize } : \textstyle\sum_{j=1}^{n} p_j x_j \\ &\text{subject to } : \textstyle\sum_{j=1}^{n} w_j x_j \leq C \\ &x_j \in \{0, 1\} \;\; \forall j \in \{1, \ldots, n\} \end{aligned} \tag{1}$$

The 0/1 knapsack problem (KP) is a well-known combinatorial optimization problem which typically arises in resource allocation having financial constraints. It occurs in several real world decision-making processes such as finding the least wasteful way to cut raw materials, selection of capital investments and financial portfolios, selection of assets for asset-backed securitization, and generating keys for the Merkle–Hellman knapsack crypto system [1]. The KP has been a problem of interest since the early days. The development of effective exact algorithms for KP started in 1970s [2–5]. Several problems can be reduced to KP [6]. Karp [7] established that KP is p-complete i.e. if one finds a polynomial time algorithm for KP then one would solve a wide range of problems in polynomial time.

Quantum Inspired Evolutionary Algorithms (QIEAs) are Evolutionary Algorithms that use concepts and principles of

```
Procedure QIEA
Begin
        t ← 0
        Initialize (Q(t)), b
        while ( t<MaxIterations)
        Begin
             t ← t+1
             make P(t) from Q(t)
             repair P(t)
             copy P(t) to B(t)
             for r from 0 to η1 do
                 for s from 0 to η2 do
                   make P(s) from Q(t)
                   repair P(s)
                   evaluate P(s)
                     for each j ∈ n if ( p_j^s better than p_j^t ) then p_j^t ← p_j^s
                end //*for s*//
                 for each j ∈ n if ( p_j^t is better than b_j^t) then b_j^t ← p_j^t
                 for each j ∈ n if (b_j^t is better than b) then b ← b_j^t , bqbit ← q_j^t
                 for each j ∈ n update q_j^t based on b_j^t
             end //*for r*//
              for each j ∈ n update q_j^t based on b
           end //*while*//
End
```

Fig. 1 Pseudo-code for QIEA

Table 1 Existing QIEA's used to solve variants of KP

Modifications	Examples	Problem type	Maximum size (n = items count; m = knapsacks count)
Original QIEA (QIEA-o)	[10,28]	KP	n = 500
Modified initialization of qubit individuals in QIEA	[27]	KP	n = 500
	[42]	KP	n = 500
	[23]	KP	n = 500
	[43]	KP	n = 500
Modification in termination criteria	[27]	KP	n = 500
Modification in attractor, gate, etc. while updating the qubit	[27]	KP	n = 500
	[31]	KP	n = 500
	[33]	DKP	n = 10,000
Modifying repair function based on domain knowledge	[42]	KP	n = 500
Replacing local and/or global migration of QIEA-o with other strategy	[44]	KP	n = 500
	[45]	KP	n = 500
Incorporation of genetic operator mutation	[33]	DKP	n = 10,000
Re-initialization of qubits	[44]	KP	n = 500
Inclusion of domain knowledge in the search process.	[33]	DKP	n = 10,000
QIEA-o with parallel implementation	[29]	KP	n = 500
QIEA-o implementation on GPU	[34]	KP	n = 250

KP simple knapsack problem, *DKP* difficult knapsack problems

quantum computing such as quantum bits (qubits), superposition of states and quantum gates. QIEAs were introduced for the first time by Narayanan and Moore [8] in the 1990s to solve the travelling salesman problem. Han and Kim [9,10] proposed a practical QIEA having the characteristics as qubit representation of individuals, Q-gate to guide the individuals towards better solutions. Han [11] showed that the inherent probabilistic mechanism of QIEAs result in good balance between exploration and exploitation. Platel et al. [12] showed that QIEAs are multi-model Estimation of Distribution Algorithms (EDA). Zhang [13] categorized QIEAs into three types viz. binary Observation (bQIEAs), real observation (rQIEAs) and QIEA like algorithms (iQIEAs) and presented a comparison to illustrate that QIEAs are better and more robust than EDAs because QIEAs can solve a broader range of problems using a smaller size of population. All QIEAs use qubit as a probabilistic representation of individuals and define a Q-gate as an evolutionary operator which guides the individuals towards generation of solutions that are better than their parents. The qubit representation has a better characteristic of population diversity than other representations [11,13] used in EAs. Zhang et al. [14] present different Q-gates with empirical performance comparison.

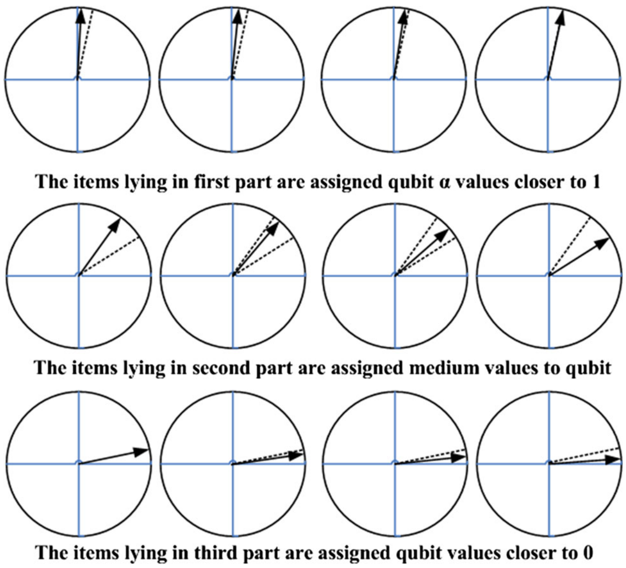


Fig. 2 Initializing qubits based on concept of greedy solution and core concept

QIEAs which are based on binary observation have been favoured most by researchers. Zhang [13] has further categorised this type of QIEAs into original bQIEA (bQIEAo) [10]; bQIEA with crossover and mutation (bQIEAcm) [15–17]; bQIEA with a novel update method for Q-gates (bQIEAn) [18]; and hybrid bQIEA (bQIEAh) [19–26].

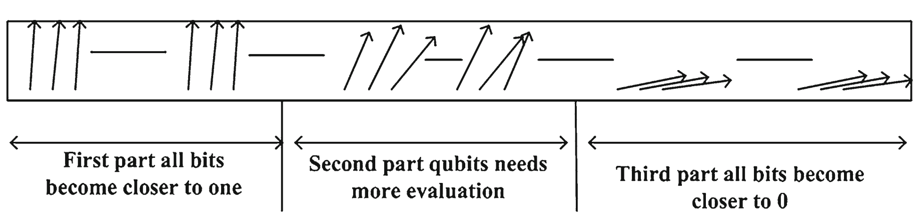
Knapsack problems have been chosen by many researchers working on different QIEAs as a suitable example to investigate the performance of their "modified" version viz., bQIEAo [10,27–32], bQIEAcm [15–17], and bQIEAh [19–24,33]. None of these implementations attempt to solve problem instances of size more than 10,000 items.

Nowotniak and Kucharski [34] implemented a sequential version of QIEA on Intel Core i7 2.93GHz CPU and a parallel version in GPU-based massively parallel computing environment (NVidia CUDA™technology) with improved rotation angles in quantum genes. Experiments on a single knapsack problem of size 250 items only have been per-

```
Procedure RepairGreedy (x)
begin
  knapsack-overfilled ← false
  weight ← Σ wjxj | j=(1,…,n)
  if (weight >  C)
  then knapsack-overfilled ← true
  while (knapsack-overfilled) do
  begin
    select ith item from the knapsack having smallest profit by weight ratio
    xi ← 0   (i.e. remove ith item from knapsack)
    weight← weight – wi
    if (weight≤ C)
    then knapsack-overfilled ← false
  end
  for each item j not in knapsack considered in decreasing order of profit by weight ratio do
  begin
   if (weight+wj ≤ C)
   begin
    xj ←1   (i.e. add jth item into knapsack)
    weight← weight +wj
   end
  end
end
```

Fig. 3 Pseudo-code for RepairGreedy

Fig. 4 State of qubits after a few iterations of evolution. Qubits are sorted such that profit by weight ratio decreases from left to right



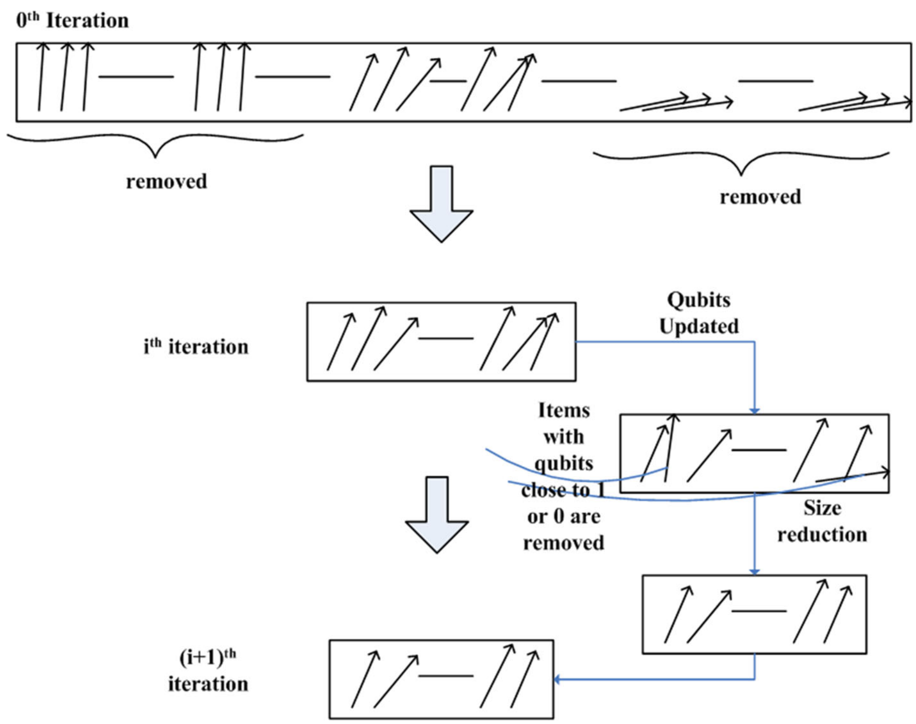


Fig. 5 Illustrating the process of size reduction during subsequent generations of QIEA

formed. The algorithm has been implemented on a system of eight GPU devices (4× Tesla T10 GPU, GTX 285, dual-GPU GTX 295 and Tesla C2070 GPU) and run to 500 generations in approximately 0.00034 s using a population of 10 quantum individuals.

An exact dynamic programming method has been implemented by Boyer et al. [35] on NVIDIA GPU architecture (NVIDIA GTX 260 graphic card with 192 cores, 1.4 GHz). The performance of the sequential implementation on Intel Xeon 3.0 GHz and has been compared with the parallel. The parallel implementation has been shown to solve randomly generated correlated knapsack problems of size up to 100,000 elements in 289.21 s.

The performance of the QIEAs has not been studied on KP instances having more than 10,000 items. Some DKP instances have been studied by Patvardhan et al. [33]. But Reilly [36] proved that these problems are similar and thus are not so difficult. In this work some DKPs are selected that have been proved to be difficult in actual practice [36–39]. An improved QIEA called QIEA-PSA (using initials of authors P-Patvardhan, S-Sulabh and A-Anand) is presented. The performance of QIEA-PSA is far better than QIEA in terms of solution quality as well as time taken for convergence on considered DKPs of size up to 290,000 variables.

Even a serial implementation of QIEA-PSA compares favorably with a parallel implementation of an exact algorithm [35]. A comparison, presented on benchmark KP instances, shows that QIEA-PSA also outperforms the modified binary particle swarm optimization algorithm recently proposed by Bansal and Deep [40].

The ideas incorporated in QIEA-PSA are applicable in general and can also be used with profit for design of better performing QIEAs for other similar and not so similar problems.

The rest of the paper is organized as follows. The types of difficult KPs considered in this work are explained in Sect. 2. QIEA is introduced in Sect. 3. The improved QIEA-PSA is presented in Sect. 4. Results of the experiments and conclusions are presented in Sects. 5 and 6 respectively.

2 Difficult knapsack problems (DKPs)

Several groups of randomly generated instances of DKPs have been constructed to reflect special properties that may influence the solution process in [37,38]. In all these instances the weights are uniformly distributed in a given interval with data range $R = 1000$. The profits are expressed as a function of the weights, yielding the specific properties of each group.

Nine groups of problems described below are specified as DKPs. The performance of QIEA on these problems has been studied in [33].

- *Uncorrelated data instances* p_j and w_j are chosen randomly in $[1, R]$. In these instances there is no correlation between the profit and weight of an item.
- *Weakly correlated instances* Weights w_j are chosen randomly in $[1, R]$ and the profits p_j in $[w_j - R/10,\ w_j + R/10]$ such that $p_j \geq 1$. Despite their name, weakly correlated instances have a very high correlation between the profit

```
Procedure QIEA-PSA
Begin
        sort the items in problem in decreasing order of their profit by weight ratio
        t ← 0, C_orginal ← C
        InitializeGreedy(Q(t))
        Initialize the best solution b as the greedy solution of the given knapsack
        set K ← Φ
        while ( t<MaxIterations and C > 0.01 * C_orginal)
        Begin
              t ← t+1
              Make P(t) from Q(t)
              RepairGreedy P(t)
              copy P(t) to B(t)
              for r from 0 to η1 do
                  for s from 0 to η2 do
                    Make P(s) from Q(t)
                    RepairGreedy P(s)
                    evaluate P(s)
                      if ( p_j^s better than p_j^t ) then p_j^t ← p_j^s
                end //*for s*//
                  if ( p_j^t is better than b_j^t) then b_j^t ← p_j^t
                  if (b_j^t is better than b) then b ← b_j^t , bqbit ← q_j^t
                  Update q_j^t based on b_j^t
             end //*for r*//
               Update q_j^t based on b
               for each item i in knapsack
               Begin
                   if (bestqbit[i]>0.9)
                   begin
                     K ← K U i
                     Reduce problem size by removing the item i
                     C ← C - w_i
                   end
                   if (bestqbit[i]<0.1)
                   begin
                     Reduce problem size by removing the item i
                   end
               end
         end
         for each item i of reduced problem
         begin
            if (b(i) is set)
                K ← K U i
         end
end
```

Fig. 6 Pseudo-code for QIEA-PSA

and weight of an item. Typically the profit differs from the weight by only a few percent.

- *Strongly correlated instances* Weights w_j are distributed in [1, R] and $p_j = w_j + R/10$. Such instances correspond to a real-life situation where the return is proportional to the investment plus some fixed charge for each project.
- *Inverse strongly correlated instances* Profits p_j are distributed in [1,R] and $w_j = p_j + R/10$. These instances are like strongly correlated instances, but the fixed charge is negative.
- *Almost strongly correlated instances* Weights w_j are distributed in [1, R] and the profits p_j in $[w_j + R/10 - R/500, w_j + R/10 + R/500]$. These are a kind of fixed-charge problems with some noise added. Thus they reflect the properties of both strongly and weakly correlated instances.
- *Subset sum* The profits and weights are same for all the items.
- *Even–odd subset sum* Weights w_j are randomly selected from the set of even numbers in [1, R].The profit p_j is same as weight.
- *Even–odd knapsack* Weights w_j are randomly selected from the set of even numbers in [1, R]. $p_j = w_j + R/10$.
- *Uncorrelated instances with similar weights* Weights w_j are distributed in [10000, 10010] and the profits p_j in [1, 100].

However, Reilly [36] observed that the induced correlation is strong in the following types: weakly correlated, almost strongly correlated, strongly correlated, subset sum, inversely strongly correlated, even–odd and even–odd subset sum problems. Therefore, all these instances are very similar.

Bounded strongly correlated has been described as another type of DKPs in [39].

Bounded strongly correlated Bounded instances are generated with w_j uniformly random in [1, 1000] and $p =$

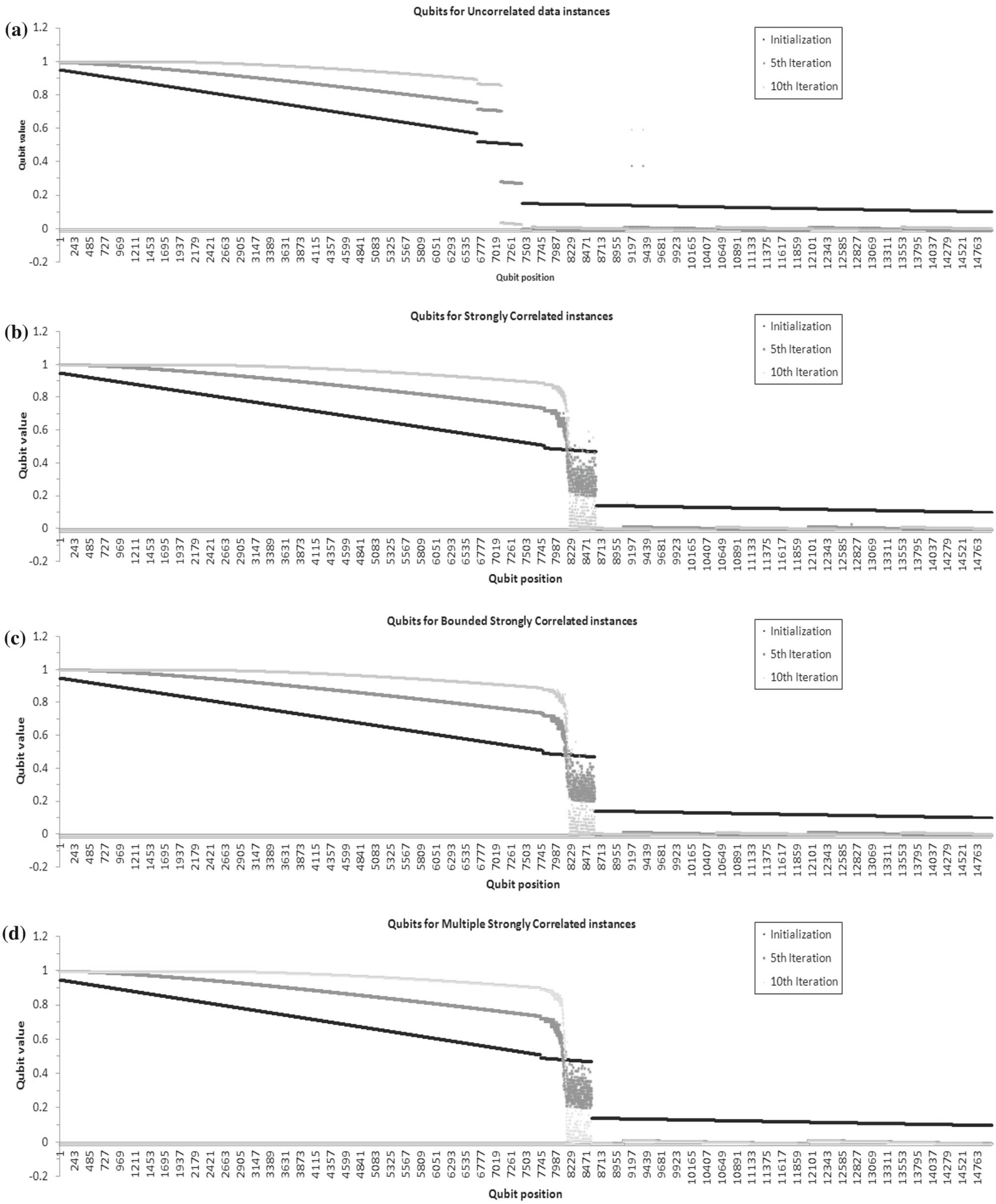


Fig. 7 Showing density of qubits on values between 0 and 1 for various types of problem instances. It presents the trend in settlement of qubits (at value close to 0 or 1) through the generations

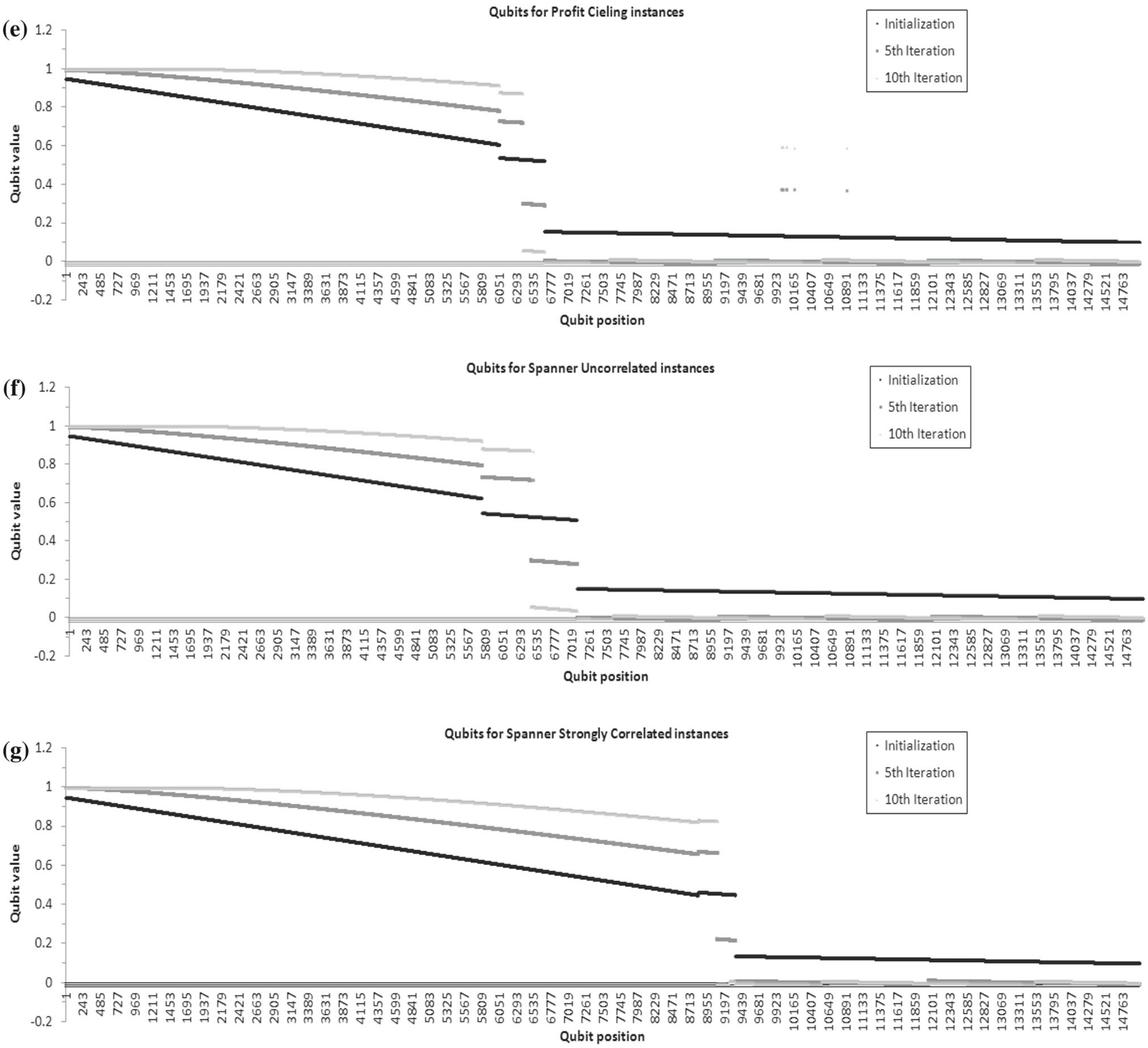


Fig. 7 continued

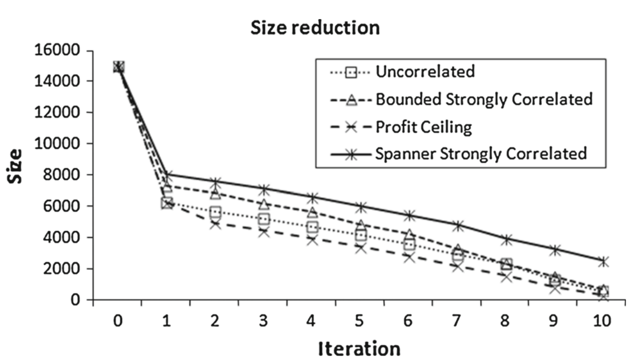


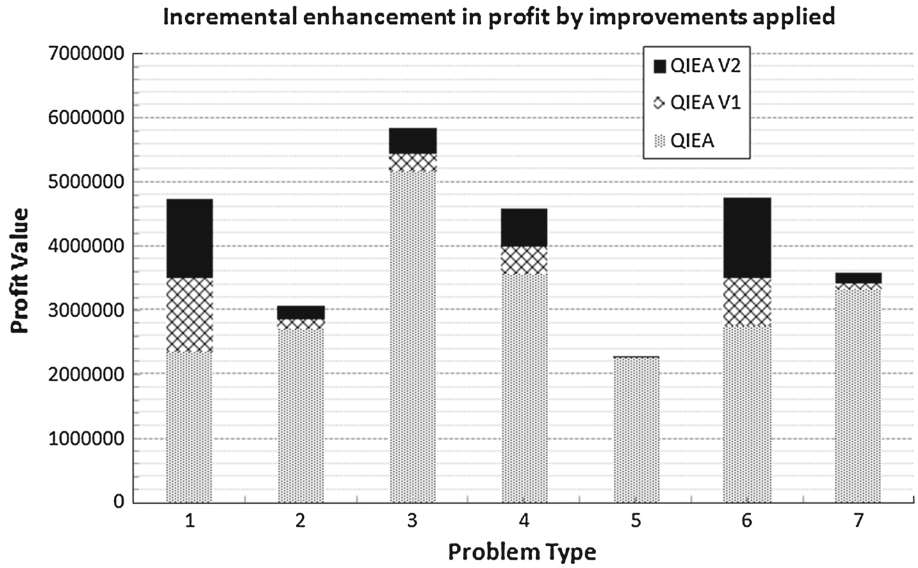
Fig. 8 Size reduction after the iteration observed for selected instances of size 15,000 items

$w_j + 100$. The bounds m_j are uniformly random in [1, 10], and the instance are transformed to a KP using the technique described in [41], until n items are present.

Pisinger [38] constructed some types of KP instances with small coefficients where state of the art exact algorithms perform badly. These are as follows.

- *Spanner instances span (v,m):* These instances are constructed such that all items are multiples of a quite small set of items—the so-called spanner set. The spanner instances span (v,m) are characterized by the following three parameters: v is the size of the spanner set, m is the multiplier limit, and finally any distribution (uncorrelated, weakly

Fig. 9 Profits added incrementally by improvements applied on simple QIEA



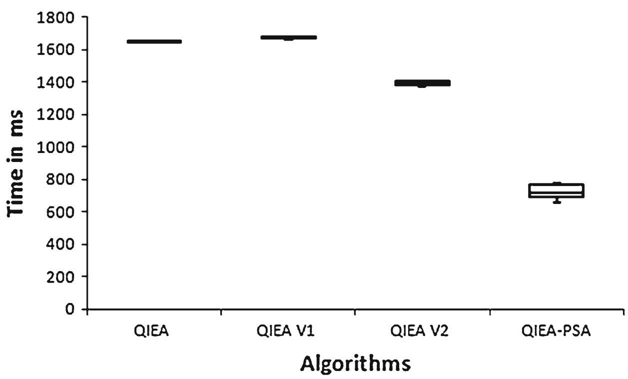


Fig. 10 Box plots showing reduction in time taken due to improvements applied on simple QIEA

Table 2 Various versions of QIEA mapped with the improvements included in them

Improvements	QIEA	QIEA V1	QIEA V2	QIEA-PSA
Sorted input and greedy repair	×	√	√	√
Greedy initialization of qubit individuals and best solution	×	×	√	√
Size reduction	×	×	×	√

correlated, strongly correlated, etc.) of the items in the spanner set may be taken. More formally, the instances are generated as follows: A set of v items is generated with weights in the interval [1, R], and profits according to the distribution. The items (p_k, w_k) in the spanner set are normalized by setting $p_k := 2p_k/m$ and $w_k := 2w_k/m$. The n items are then constructed, by repeatedly choosing an item (p_k, w_k) from the spanner set, and a multiplier is randomly generated in the interval [1, m]. The constructed item has profit and weight (a * p_k; a * w_k).

- *Multiple strongly correlated instances mstr*($\mathbf{k_1}$, $\mathbf{k_2}$, $\mathbf{d}$): These instances are constructed as a combination of two sets of strongly correlated instances. Both instances have profits pj := wj + ki where k_i, i = 1, 2 is different for the two instances.
  The multiple strongly correlated instances mstr(k_1, k_2, d) are generated as follows: the weights of the n items are randomly distributed in [1, R]. If the weight w_j is divisible by d, then set profit to $p_j := w_j + k_1$ otherwise set it to $p_j := w_j + k_2$.
  According to Pisinger difficult instances could be obtained with the parameters mstr(3R/10, 2R/10, d). Choosing d = 6 results in the most difficult instances, but values of d between 3 and 10 can all be used.
- *Profit ceiling instances pceil(d):* These instances have the property that all profits are multiples of a given parameter d. The weights of the n items are randomly distributed in [1, R], and the profits are set to $p_j = d\left\lceil w_j/d \right\rceil$. Pisinger has experimentally found that taking d = 3 yields sufficiently difficult instances.

Only sufficiently different and difficult problem instances of KP are considered in this paper. These are as follows.

Type 1. Uncorrelated data instances
Type 2. Strongly correlated instances
Type 3. Bounded strongly correlated instances
Type 4. Multiple strongly correlated instances: mstr(3R/10, 2R/10, 6).
Type 5. Profit ceiling instances: pceil(3)
Type 6. Spanner Instances: uncorrelated span(2,10)
Type 7. Spanner Instances: strongly correlated span(2, 10).

Table 3 Average profits obtained using different versions of QIEA

Algo. version	Problem type						
	1	2	3	4	5	6	7
QIEA	2,358,523	2,714,996.5	5,175,234.8	3,569,376.5	2,254,784.4	2,764,582.9	3,323,778.7
QIEA V1	3,495,745.1	2,858,587	5,452,654.6	3,992,098.2	2,257,710	3,499,206.3	3,414,674.4
QIEA V2	4,744,184.6	3,070,425.8	5,847,073.4	4,594,496	2,260,612.5	4,768,736.4	3,587,039.4
QIEA-PSA	4,744,182.1	3,070,425.5	5,847,073.7	4,594,495.8	2,260,612.5	4,768,734.7	3,587,031.4

Table 4 Average computing times (ms) observed using different versions of QIEA

Algo. version	Problem type						
	1	2	3	4	5	6	7
QIEA	1648.9	1645	1645.9	1644.9	1644.6	1644.5	1646.2
QIEA V1	1674.3	1678.7	1669.9	1672.9	1663.4	1667.7	1671.5
QIEA V2	1399.5	1406.7	1410.7	1410.3	1372.1	1374.5	1389.6
QIEA-PSA	718.5	772.8	771.3	774.1	656.5	669.7	714.2

Table 5 Results for uncorrelated instances

Size (K)	Profit			Time		
	QIEA	QIEA-PSA	Gap %	QIEA (ms)	QIEA-PSA (ms)	Gap %
10	1,592,880.5	3,157,580.4	49.55455	1091.2	469.6	56.96218397
30	4,648,369	9,487,718	51.00634	3263.6	1422.3	56.41923198
50	7,699,781.2	15,812,131.6	51.3043	5442.2	2374	56.37753444
70	10,729,274.2	22,127,795.9	51.51231	7596.4	3332.2	56.13451813
90	13,757,594.3	28,446,200.2	51.63643	9763.7	4281.2	56.15182373
110	16,790,093.6	34,769,640.6	51.71051	11,964.5	5231.8	56.27074034
130	19,809,372.3	41,089,609.3	51.78986	14,104.7	6265.5	55.57866982
150	22,833,427.6	47,401,780.4	51.83004	16,295.7	7129.9	56.24670735
170	25,847,625.1	53,730,282	51.89376	18,684.5	8078.2	56.76345167
190	28,886,369.9	60,058,768	51.90318	20,674.1	9043.7	56.25579099
210	31,904,191.8	66,365,561.3	51.92658	22,869.5	9997.6	56.28397096
230	34,915,549.5	72,678,697.6	51.95905	25,095.2	10,942.5	56.39598358
250	37,935,471.6	79,003,708.7	51.98267	27,403.2	11,841.5	56.78780052
270	40,946,787.8	85,337,687.5	52.01793	29,631.9	12,801.1	56.79939625
290	43,970,453.6	91,659,492.4	52.0285	31,930.6	13,787.2	56.82127697

3 Quantum-Inspired Evolutionary Algorithm (QIEA)

QIEA uses quantum bits (qubits) as the smallest unit of information for representing individuals. Each qubit is represented as $q_i = [\alpha_i\, \beta_i]^T$ where α_i and β_i are complex numbers representing probabilistic state of qubit 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 $|\alpha_i|^2 + |\beta_i|^2 = 1$. When a qubit is observed, the result fall into [0, 1] depending on the probability defined by the qubit. For QIEA, α_i and β_i are considered real without losing the generality. Han and Kim [10] presented a QIEA where Q(t) is the qubit population, P(t) is the population of individual solutions, B(t) is the set of best solutions corresponding to each individuals, C is the capacity of the knapsack. Elements of Q(t) are initialized to value $1/\sqrt{2}$. Subsequently P(t) and B(t) are initialized by observing the qubits of Q(t). The algorithm then iterates through following steps till termination criterion is met:

- update Q(t) when the individual observed in P(t) does not improve over the local best or global best individual in B(t).
- observe Q(t) to form new P(t),
- select new best population B(t)

A rotational Q-gate is used for updating the quantum bits in Q(t). The qubit i in an individual of Q(t) is rotated by angle $\Delta\,\theta_i$, proposed as 0.01π, either clockwise or anti-clockwise depending on the corresponding bit in best individual is either 0 or 1. Han and Kim [10] have solved strongly correlated knapsack problem using QIEA and show that the best variant of QIEA converges within maximum of 1000 generations which takes approx. 0.1 sec to solve problem of size 500 items. No comparison with optimal solution is reported.

Table 6 Average time and average FES taken to compute best solution for uncorrelated instances

Size (K)	QIEA		QIEA-PSA	
	Time	FES	Time	FES
10	1019.7	234.1	226.2	121.3
30	3048.6	234	8	2
50	5157.9	237.4	13.5	2
70	6672.7	220.2	19.8	2
90	8725.2	223.8	25.3	2
110	11,104.1	232.5	31.2	2
130	13,525.4	240.2	37.8	2
150	14,981.8	230.4	43.1	2
170	16,317.7	218.9	49	2
190	18,626.8	225.8	55.1	2
210	20,593.8	225.6	61.6	2
230	23,816.4	237.7	67.5	2
250	26,277.8	240.4	77.4	2
270	27,744.5	234.7	84.1	2
290	29,715.2	233.1	90.2	2

The QIEA implementation used in the present work is given in Fig. 1. The notations used are as follows.

Q(t): Qubit 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.
n: Number of items to be considered in problem.
t: the current iteration.

Qubits in Q(t) are initialized with $1/\sqrt{2}$ and best binary solution b with zeroes *(lines 1 and 2)*.
The QIEA iterates Maxiterations times through the remaining tasks described in *lines 4–24*.
In every iteration individuals of Q(t) are observed and the solutions are repaired to make them feasible *(lines 7 and 8)*.
B(t) and P(t) are initialized by these feasible solutions *(line 9)*.
The following tasks are then repeated η_1 times *(lines 10–22)*.

- The qubits 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 11–18)*.
- Individual at every position in B(t) is replaced by corresponding individual in P(t) if found better *(line 19)*.
- The best solution from among b and B(t) is moved into b *(line 20)*.

Table 7 Results for strongly correlated instances

Size (K)	Profit			Time		
	QIEA	QIEA-PSA	Gap %	QIEA (ms)	QIEA-PSA (ms)	Gap %
10	1,814,672	2,048,052	11.39531	1087.6	518	52.3726
30	5,423,535	6,143,630	11.72104	3264.3	1552.2	52.44873
50	9,033,409	10,242,349	11.80344	5448	2586.3	52.52665
70	12,636,373	14,336,783	11.86051	7582.9	3615.9	52.31469
90	16,243,048	18,430,675	11.86957	9771.1	4701.3	51.8854
110	19,843,458	22,524,755	11.90381	11,972.4	5722.5	52.20169
130	23,442,926	26,616,784	11.92429	14,111.2	6858.9	51.394
150	27,041,275	30,709,519	11.94498	16,307.4	7843.1	51.90442
170	30,652,004	34,806,943	11.9371	18,596.8	8875.4	52.27338
190	34,247,899	38,898,425	11.95556	20,685.4	9965	51.82613
210	37,844,768	42,987,555	11.96343	22,865.3	11,028	51.76961
230	41,452,441	47,085,228	11.96297	25,084	12,068	51.88969
250	45,048,621	51,177,553	11.97583	27,380	13,055.3	52.31756
270	48,647,077	55,269,189	11.98157	29,702.8	14,129	52.43198
290	52,258,454	59,369,020	11.9769	31,918.2	15,221.5	52.31066

- *Local update:* Each individual in Q(t) is updated based on the corresponding best solution in B(t)*(line 21)*.

Each iteration restarts after updating the individuals in Q(t) globally *(line 23)* based on the best solution observed so far.

Several attempts have been made to solve variant(s) of KP using QIEAs. Table 1 lists various attempts to design QIEAs in literature to solve KP instances.

Table 8 Time and FES taken to compute best solution for strongly correlated instances

Size (K)	QIEA		QIEA-PSA	
	Time	FES	Time	FES
10	1005	231.5	352.2	170.5
30	3015.7	231.5	1307	211
50	5104	234.8	2264.7	219.4
70	6988.7	231	3206.4	222.3
90	8830.8	226.5	3860.7	206.4
110	10,888.7	228	5138.4	224.7
130	13,218.8	234.8	6336.2	231.5
150	14,969.9	230	7228.3	230.8
170	17,464.8	235.4	8143.6	229.9
190	18,956.8	229.6	9041.2	227.4
210	20,655.4	226.3	10,036.5	228
230	24,053.5	240.3	11,145	231.2
250	24,434.3	223.6	12,116.2	232.7
270	28,227.2	238.2	12,949.5	229.7
290	29,515.9	231.6	14,016.2	230.6

4 QIEA-PSA

Several new ideas have been incorporated in QIEA-PSA for enhanced performance. Details are as follows.

I. *Simplified Input* The items are sorted in the decreasing order of their profit by weight ratio so that the items can be assigned the probabilities of their inclusion in the solution accordingly in order of decreasing probabilities.

Table 10 Time and FES taken to compute best solution for bounded strongly correlated instances

Size (K)	QIEA		QIEA-PSA	
	Time	FES	Time	FES
10	1036.1	238.5	367	179.2
30	3031	229.1	1349.9	218.3
50	4915.6	226.4	2343.9	224.4
70	6966.4	229.8	3349.7	230.2
90	8794.2	226	4294.1	228.4
110	11,310.8	235.9	4729.1	205.8
130	12,828.6	227.7	6299.3	230
150	15,247.1	234.1	7235.2	230.1
170	16,964.4	230.1	8373.3	235
190	18,823	228.2	9145.3	229
210	21,404.5	234.5	10,171.5	230.1
230	23,587.7	235.6	11,229	232.3
250	25,748	236.1	11,895.9	227.2
270	27,585.9	232.8	12,973.9	229.4
290	29,651.8	233.1	14,018.3	229.9

Table 9 Results for bounded strongly correlated instances

Size (K)	Profit			Time		
	QIEA	QIEA-PSA	Gap %	QIEA (ms)	QIEA-PSA (ms)	Gap %
10	3,450,445.2	3,891,364.5	11.33241761	1088.2	514	52.76545428
30	10,322,412.7	11,678,451.2	11.61187546	3316.6	1549.3	53.281786
50	17,141,647.4	19,434,364	11.79724826	5441.9	2618.2	51.88847058
70	23,976,306.3	27,191,182.2	11.82347831	7598.8	3645	52.03185893
90	30,804,942.7	34,949,992.8	11.86010355	9760.1	4710	51.74205823
110	37,657,680	42,734,418.1	11.87991752	12017.5	5773.5	51.95512079
130	44,498,663.1	50,500,644.5	11.885104	14,118.8	6860.8	51.40604771
150	51,322,678.3	58,264,346.9	11.91419878	16,322.8	7875.1	51.75305479
170	58,160,564.7	66,033,115.4	11.92219697	18,476.8	8932.3	51.65647091
190	64,953,443.3	73,770,442.7	11.95200214	20,686.4	10,009.9	51.61121828
210	71,817,548.7	81,557,575	11.94256899	22,882.7	11,074	51.60525891
230	78,640,981.9	89,316,793.9	11.9528028	25,073.8	12,114.7	51.68367612
250	85,497,539.5	97,099,796.8	11.94882427	27,316.8	13,121.2	51.9664945
270	92,314,713.4	104,848,407.9	11.95412868	29,682.6	14,178.1	52.23401936
290	99,136,949.7	112,604,462.5	11.96003323	31,877.9	15,270.4	52.09700475

Table 11 Results for multiple strongly correlated instances

Size (K)	Profit			Time		
	QIEA	QIEA-PSA	Gap %	QIEA (ms)	QIEA-PSA (ms)	Gap %
10	2,389,952	3,063,951	21.99757	1087.1	517.2	52.42373
30	7,116,975	9,193,808	22.58945	3258.8	1562.8	52.04377
50	11,839,629	15,327,529	22.75582	5445.1	2589.4	52.44473
70	16,548,073	21,456,511	22.87624	7589.8	3648.3	51.93151
90	21,258,698	27,583,342	22.92926	9773.5	4708.8	51.82063
110	25,967,738	33,710,530	22.96848	12,006.2	5760.1	52.02272
130	30,674,366	39,838,049	23.00235	14,143.6	6802.6	51.90292
150	35,377,325	45,964,429	23.03326	16,319.8	7861.3	51.82972
170	40,090,334	52,095,172	23.04405	18,475.3	8933.5	51.64627
190	44,791,399	58,219,343	23.0644	20,665.6	9982.4	51.69547
210	49,493,808	64,345,406	23.08106	22,905.7	11,033.6	51.83027
230	54,203,261	70,473,608	23.08715	25,148.1	12,098.9	51.88928
250	58,912,471	76,601,066	23.09185	27,335.4	13,084.1	52.13499
270	63,608,767	82,725,872	23.10899	29,679.1	14,160.2	52.28886
290	68,320,024	88,859,579	23.11462	31,831.6	15,244.4	52.10903

II. *Initialising the qubits using a good heuristic* The qubits are collapsed to 0 or 1 state on observation with probabilities defined by $|\alpha_i|^2$ and $|\beta_i|^2$ respectively. They thus reflect the probability of the corresponding item being included in the solution. The qubits are initialized to values decreasing from 0.95 to 0.2 for items sorted in the decreasing order of the profit by weight ratios in such a way that the list of items is divided into three parts where first and third parts have qubits closer to 1 and 0 respectively while the second part contains qubits of value around $1/\sqrt{2}$ (Fig. 2). This implies that elements in second portion take part in the QIEA-PSA evolution process for a longer duration. Elements in the first and third parts are removed early as a result of size reduction explained in modification number V.

III. *Using a known good solution as the initial best solution for further improvement* The best solution b is initialized to the greedy solution of the problem. Till some better solution is found the qubits are rotated towards this best solution.

IV. *Modifying the repair to speed up the evolution* Repair is used to correct the solution after Q-bits are collapsed to make the resultant solution feasible. In the repair step of the QIEA-PSA, named RepairGreedy, the solution is improved apart from making it feasible. When capacity constraint is violated, items are removed from knapsack in such a way that the items with lesser profit by weight ratio are chosen. Similarly an item with larger profit by weight ratio is chosen, when required to fill in the knapsack. Figure 3 presents the pseudo-code for RepairGreedy.

V. *Size reduction* QIEA keeps rotating the qubits of individuals towards the best solution it finds during evolution. Due to this property the qubits of items lying in the first part gradually acquire values very close to 1 and that of items in third part acquire values closer to zero as illustrated in Fig. 4. At this point, the probability of rotating these qubits in the opposite direction during further evolution becomes almost negligible. Continuing with such qubits in subsequent iterations is futile.

Table 12 Average time and average FES taken to compute best solution for multiple strongly correlated instances

Size (K)	QIEA		QIEA-PSA	
	Time	FES	Time	FES
10	965.7	222.5	383.2	185.6
30	3057.8	235	1385.4	222
50	5216.7	240.1	2354.4	227.9
70	6765.5	223.3	3380	232.3
90	9037.5	231.7	4326.3	230.3
110	10,945.9	228.5	5212.7	226.9
130	12,958	229.6	6396.1	235.6
150	15,427.8	236.7	5880.2	188
170	17,640.5	239.2	6712.6	188.8
190	18,291	221.7	7422.1	186.9
210	21,822	238.6	9202.2	209.2
230	22,357.8	222.9	10,096.8	209.5
250	25,480.3	233.5	10,850.3	208.1
270	28,010.4	236.4	10,621.9	188.4
290	29,976.6	235.9	12,892.2	212.3

Table 13 Results for profit ceiling instances

Size (K)	Profit			Time		
	QIEA	QIEA-PSA	Gap %	QIEA (ms)	QIEA-PSA (ms)	Gap %
10	1,504,390	1,508,226	0.254329	1082.9	437	59.64515
30	4,511,274	4,523,104	0.261546	3261.1	1307.9	59.8925
50	7,521,739	7,541,601	0.263373	5449.9	2179.3	60.0113
70	10,527,660	10,555,613	0.264811	7589.2	3059.1	59.6914
90	13,534,130	13,570,181	0.265668	9795.1	3924.6	59.93262
110	16,540,128	16,584,208	0.265797	11,974.2	4795.1	59.95354
130	19,543,200	19,595,383	0.266306	14,114.6	5666.3	59.85502
150	22,547,705	22,607,951	0.266485	16,385.9	6538	60.0934
170	25,557,020	25,625,361	0.266695	18,483.3	7414.8	59.88374
190	28,560,587	28,637,065	0.267057	20,664	8295.3	59.85616
210	31,560,192	31,644,801	0.267371	22,879.8	9162.3	59.95442
230	34,571,342	34,664,006	0.267323	25,108.2	10,047.9	59.98141
250	37,574,880	37,675,666	0.267511	27,435.5	10,874.4	60.3634
270	40,577,519	40,686,350	0.267488	29,711.5	11,755.5	60.43435
290	43,590,190	43,707,058	0.267388	31,780.8	12,672.8	60.12432

The qubit individual 'bqbit', stores the qubit string that generates the best individual on collapse. When the qubits in bqbit cross the stipulated thresholds, the corresponding items are permanently taken as being selected (for qubit value >0.9) or being outright rejected (for qubit value <0.1) and hence removed from the search process. Figure 5 illustrates this reduction process.

VI. *Re-initializing population of local best solutions* In QIEA-PSA the population of best solutions is reinitialized whenever the size of the problem is reduced. This is required because the problem changes after the size reduction. This is also beneficial in improving the exploration capability of the algorithm as it increases the diversity in the population.

The pseudo-code for QIEA-PSA is presented in the Fig. 6. The set of integers in range [1..n], K represents the knapsack solution, such that $i \in$ K if ith item is included in solution. The notation is the same as used in Sect. 3.

The maximum number of iterations in algorithm is controlled using a global constant, MaxIterations. When using size reduction, the maximum number of iterations, till size of the problem reduces to 10 % of the original, can be found easily for a set of problems. The size of population in the presented QIEA-PSA, can be chosen based on the difficulty of instance to be solved.

The algorithm starts by sorting the input in descending order of profit by weight ratio *(lines 2 and 3)*. Qubits in Q(t) are initialized as described earlier and best binary solution b with greedy solution *(lines 4 and 5)*. It iterates Maxiterations times (or till the capacity of reduced problem is to less than 1 percent of original capacity), through the tasks described in the *lines 8–36*. In every such iteration individuals of Q(t) are observed and the solutions are repaired using the Repair-Greedy function to make them feasible *(lines 10 and 11)*. Size reduction is carried out in *lines 23–35*. Finally the solution is formed *(lines 37–41)*. If the problem size after several reductions becomes very small (line 7) (capacity of reduced knapsack is smaller than 0.01*C_{orginal}) the evolution terminates.

Table 14 Average time and average FES taken to compute best solution for profit ceiling instances

Size (K)	QIEA		QIEA-PSA	
	Time	FES	Time	FES
10	1004.5	232.4	2	2
30	2894	222.2	5.7	1.7
50	5062.2	232.5	9.8	2
70	7160.7	236.3	14.5	2
90	8951.2	228.9	19.1	2
110	10,740.9	224.7	22.9	2
130	13,267.4	235.5	27.1	2
150	15,513.8	237	31.4	2
170	16,288.8	220.8	35.3	2
190	18,111	219.8	39.8	2
210	21,053.3	230.5	43.1	2
230	23,002.6	229.5	48	2
250	25,489.4	232.8	59	2
270	25,845.5	217.9	63.9	2
290	29,275.7	230.8	68.8	2

Table 15 Results for Spanner uncorrelated instances

Size (K)	Profit			Time		
	QIEA	QIEA-PSA	Gap %	QIEA (ms)	QIEA-PSA (ms)	Gap %
10	1,857,567	3,178,895	33.92462	1083.5	444	59.02835
30	5,458,867	9,543,950	34.95158	3254.8	1333.6	59.02794
50	9,079,506	15,912,516	35.07301	5453.1	2217.3	59.35265
70	12,668,283	22,255,737	35.20911	7589.7	3115.6	58.95198
90	16,228,552	28,607,802	35.37134	9840.8	4007.3	59.28238
110	19,826,800	34,956,123	35.38168	11,959.6	4884.2	59.16397
130	23,405,716	41,302,860	35.44491	14,109.1	5769.8	59.10415
150	26,956,965	47,650,647	35.50569	16,419.1	6658.8	59.41781
170	30,523,690	53,987,020	35.54556	18,479.1	7551.6	59.1305
190	34,082,085	60,321,704	35.57091	20,674.9	8440.1	59.18256
210	37,636,247	66,692,691	35.6174	22,867.6	9323.4	59.23339
230	41,201,577	73,042,256	35.63414	25,135.7	10,225.2	59.32259
250	44,757,508	79,400,409	35.66452	27,386.4	11,059.7	59.61763
270	48,317,001	85,742,976	35.68197	29,726.6	11,962.1	59.7672
290	51,905,431	92,100,412	35.67552	31,844.7	12,876.4	59.56585

Table 16 Average time and average FES taken to compute best solution for spanner uncorrelated instances

Size (K)	QIEA		QIEA-PSA	
	Time	FES	Time	FES
10	969.3	224.1	2	1.7
30	2952	227.3	5	1.3
50	4986.5	229	9	1.6
70	7162.7	236.4	13.1	1.7
90	9162.6	233.3	16.4	1.6
110	10,928.7	229	20.2	1.7
130	13,035.1	231.4	24.2	1.7
150	15,365.8	234.5	28	1.7
170	17,212.4	233.4	31.5	1.7
190	18,909.5	229.1	35.2	1.7
210	20,999.5	230.2	39.5	1.7
230	22,206.9	221.4	43.5	1.7
250	24,699.6	226.1	49.2	1.7
270	25,958.1	218.8	369.4	9.1
290	29,162.1	229.6	471.7	10.8

5 Results and discussion

QIEA-PSA has been tested with population sizes 5, 10, 20 and 30 and MaxIterations as 5, 10 and 20. The best combination of parameters found experimentally is population size 5 and MaxIterations 10. The value of each η_1 and η_2 is empirically set to 5. Due to initialization with a good greedy solution and subsequent size reduction, executing QIEA-PSA with larger populations or for more than 10 iterations has not resulted in much gain in profit as compared to increase in computation cost for the simpler instances (those considered up to Sect. 5.3). A larger population is taken for benchmark problem instances considered in Sect. 5.4.

All of the experiments are done on Red Hat Linux(RHEL6) operating system running on Intel®Xeon®Processor E5645 having specifications as 12M Cache, 2.40 GHz, 5.86 GT/s Intel®QPI. Problems are generated using “Advanced Generator” [46]. The parameter R is taken as 1000 and the capacity of the knapsack is chosen to be 30 percent of the sum of the weights of all the items in the problem.

Sample runs on randomly generated problems of size 15,000 items of each type selected in Sect. 2 are used for presentation of the results in this section.

The experiments conducted to study the effects of various modifications introduced in QIEA-PSA are discussed in Sect. 5.1. Results with discussion for various types of KP instances considered here (mentioned as Type 1 to Type 7 in Sect. 2) are presented in Sect. 5.2. The results on instances used by Boyer et al. [35] are compared in Sect. 5.3. The comparison with modified binary particle swarm optimization of Bansal and Deep [40] is presented in Sect. 5.4.

5.1 Effect of various modifications in QIEA

Figure 7a–g shows the status of each qubit at initialization and their status after 5th and 10th iterations for various types of problem instances. The qubit number (numbers ranging from 1 to 15,000 representing the corresponding item in the sorted order according to p_i/w_i) is depicted on the x-axis whereas the qubit value is on the y-axis. The darkest line is

Table 17 Results for Spanner strongly correlated instances

Size (K)	Profit			Time		
	QIEA	QIEA-PSA	Gap %	QIEA (ms)	QIEA-PSA (ms)	Gap %
10	2,213,823	2,387,059	7.45602	1083.5	474.8	56.17808
30	6,646,099	7,178,918	7.616324	3270.9	1422.2	56.52304
50	11,065,365	11,962,272	7.699963	5431.7	2380.7	56.17457
70	15,491,982	16,744,779	7.679383	7593.3	3327.2	56.18478
90	19,918,379	21,538,025	7.716908	9802.4	4268.7	56.45757
110	24,352,913	26,334,172	7.719347	11,952.3	5243.6	56.12459
130	28,764,847	31,108,665	7.730174	14,114	6160.3	56.35519
150	33,185,424	35,892,919	7.740749	16,320.7	7109.7	56.44658
170	37,601,411	40,672,217	7.746722	18,486.5	8051.9	56.44568
190	42,020,873	45,456,137	7.753551	20,690.6	9005.9	56.47209
210	46,444,466	50,237,735	7.745682	22,888.8	9955.9	56.50334
230	50,856,682	55,014,811	7.753631	25,100	10,841.5	56.81114
250	55,277,842	59,801,699	7.76065	27,343.6	11,793	56.87636
270	59,687,080	64,576,576	7.767421	29,690.1	12,766.7	56.99554
290	64,110,863	69,362,999	7.768457	31,883.5	13,728.5	56.94495

Table 18 Average time and average FES taken to compute best solution for spanner strongly correlated instances

Size (K)	QIEA		QIEA-PSA	
	Time	FES	Time	FES
10	985.2	227.8	16.3	10.9
30	2992.8	229.2	99.1	21
50	4962	228.8	168.5	21.4
70	7180.9	237	169.5	15.7
90	8238.1	210.7	115.1	8.5
110	10,177.3	213.5	271	15.7
130	13,078.9	232.2	187.1	9.2
150	15,139.1	232.4	219.8	9.8
170	17,158.3	232.6	652.4	23.4
190	17,763.2	215.2	237.7	8.1
210	20,999.3	229.8	606.5	17.8
230	23,144.9	231.1	675.1	19.2
250	24,391.8	223.6	1007	26
270	28,083.9	237.1	722	17.5
290	30,645.4	240.8	851.8	19.1

for the initialization and the lightest line for the status after the 10th iteration.

The following points can be discerned.

(i) In all the cases the qubits initialized close to 1 (close to 0) move further towards 1 (0).
(ii) In case of strongly correlated, bounded strongly correlated and multiple strongly correlated type instances the qubits go through considerable turbulence (being updated towards 0 and towards 1 several times) before they settle. This is evidenced by the patches in the middle portion of the graph. This also indicates that better solutions are found several times during the search in which many items get changed again and again.
(iii) In other cases the qubits move relatively smoothly towards their respective final positions within a few iterations.

Figure 8 illustrates the effect of size reduction on various types of DKP instances. It also justifies the choice of 10 for MaxIterations as the problem size reduces drastically by the 10 iteration.

As described in Sect. 4, several modifications have been made in the simple QIEA to obtain QIEA-PSA. Table 2 specifies the various versions of QIEA formed along the way.

The development of algorithm starts from basic QIEA, QIEA V1 is considered as the second step followed by QIEA V2 and finally QIEA-PSA.

Tables 3 and 4 shows the average values of profits and computing times observed over 10 randomly generated problem instances of each type of size 15,000 items.

Figure 9 shows the incremental benefit in the quality of solution with incorporation of each improvement using the stacked bar graphs for different types of problems. Different types of problem considered here are listed at the end of Sect. 2. The size reduction does not affect the quality of solution. Therefore, profit values provided by QIEA-PSA and QIEA V2 are same and not plotted separately.

Figure 10 shows the reduction of time taken by the algorithm due to each improvement incorporated using the box-plot diagrams.

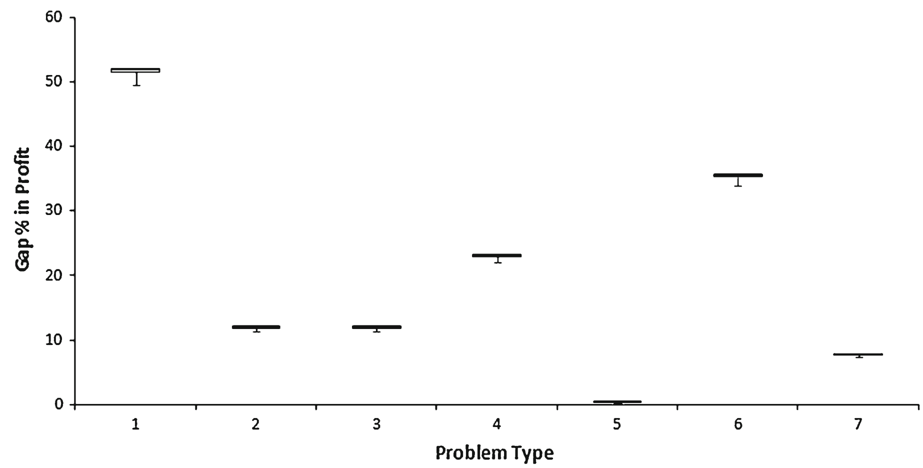


Fig. 11 Box plots comparing gap % in Profit between QIEA and QIEA-PSA for various types of problems

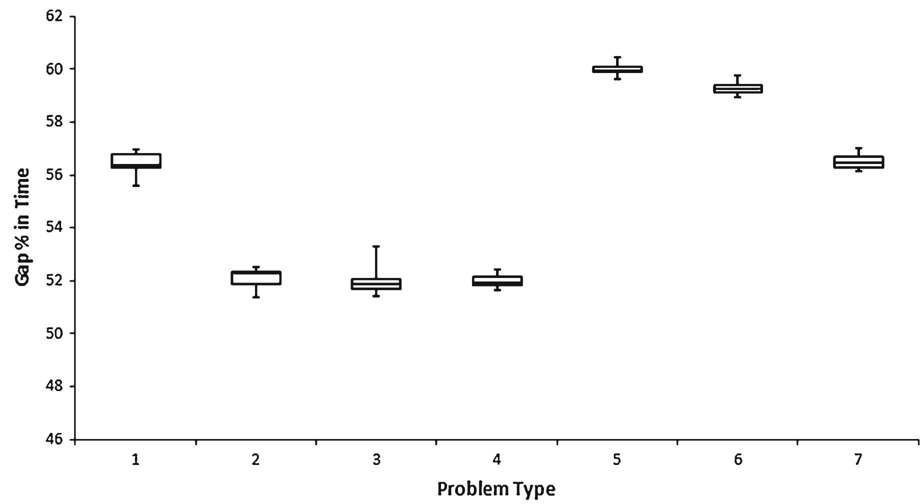


Fig. 12 Box plots comparing the gap % in Time between QIEA and QIEA-PSA for various types of problems

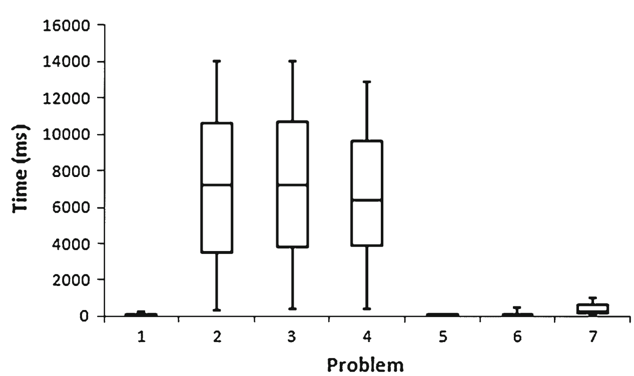


Fig. 13 Box plots showing range of time taken on average to compute best solution for all problems by QIEA-PSA

The following points emerge from Tables 3 and 4 and Figs. 9 and 10:

(i) All of the ideas of modification viz., sorted input, greedy repair, greedy initialization of qubit individuals and best solution contributed to improvement in the quality of the solutions obtained. The size reduction does not affect the quality of solution.
(ii) The size reduction results in a major reduction of time taken by QIEA-PSA.
(iii) Initialization of qubit individuals and best solution based on greedy heuristic contributed to reduction in time taken by algorithm, apart from improving the quality of solution.
(iv) QIEA-PSA provides substantially better results than QIEA on all problem types.
(v) It does so in substantially lesser times than QIEA (less than 50 % of the times required by QIEA).

QIEA V1 is very similar to enhanced QIEA of Patvardhan et al. [33]. Thus, QIEA-PSA provides better results than existing QIEAs.

5.2 Results and discussion for various types of DKP instances

The results for different types of DKP instances as listed at the end of Sect. 2 (Type 1 through Type 7) are presented for problem sizes ranging between 10,000 and 290,000 in Tables

Table 19 Time taken by QIEA-PSA till termination for problems of size 15,000

Problem	QIEA-PSA Time (ms) for size 15K	Rank
1. Uncorrelated instances	718.5	2
2. Strongly correlated instances	772.8	1
3. Bounded strongly corr. instances	771.3	1
4. Multiple strongly corr. instances	774.1	1
5. Profit ceiling instances	656.5	3
6. Spanner instances: uncorr.	669.7	2
7. Spanner instances: strongly corr.	714.2	2

Ranking of the problems based on the time taken by QIEA PSA to compute best solution

Table 20 Comparison of results on instances [47] used by Boyer et al. [35] with QIEA-PSA

Size	Avg. gap %	Computing times (s)	
		QIEA-PSA	Boyer et al.
10,000	7.37994E−05	0.6461	3.06
20,000	3.51194E−05	1.2851	11.97
30,000	5.73257E−05	1.9232	26.57
40,000	1.57768E−05	2.5592	47.43
50,000	2.24344E−05	3.2042	73.55
60,000	2.74802E−05	3.8638	105.93
70,000	1.30232E−05	4.4801	143.98
80,000	4.29929E−05	5.1528	183.15
90,000	1.75213E−05	5.8065	238.57
100,000	2.52493E−05	6.4328	289.21

5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18. Average of profit and computing times taken till termination for 10 randomly generated instances are shown in Table 5, 7, 9, 11, 13, 15, 17. The profit and computing times obtained using QIEA-PSA are compared with QIEA. Gap % between two values of profit (obtained using QIEA and QIEA-PSA) or two values of computing times is calculated as

$$\text{gap }\% = (|\text{value1} - \text{value2}|/\text{Max}(\text{value1}, \text{value2})) * 100$$

Average of gap % (over 10 instances) between profits obtained from QIEA and QIEA-PSA and average of gap % between time taken by QIEA and QIEA-PSA is also presented for problems of size from 10K to 290K. The average of time and function evaluations (FES) taken to compute best solution for 10 randomly generated instances are shown in Tables 6, 8, 10, 12, 14, 16, 18.

The box-plots in Fig. 11 present the comparison of gap % in profit observed between QIEA and QIEA-PSA for different problem types based on results in Tables 5–18. Similarly the box plots of Fig. 12 present the comparison of gap % in time taken for different problem types.

A large gap % in profits indicates that QIEA-PSA found much better solution than QIEA. This is because QIEA found solution far from optimal and so QIEA-PSA had considerable scope for improvement. Further a larger gap % in time indicates that QIEA-PSA converged much faster than QIEA either due to faster setting of qubits or due to quick reduction in size.

Following observations are made from the results in Tables 5–18.

- QIEA-PSA outperforms the simple QIEA in terms of the quality of solutions and time taken to compute them for all types of KP instances considered.
- The problem types considered are ordered in decreasing order of gap % in solution quality as follows.

  (i) Uncorrelated instances.
  (ii) Spanner uncorrelated instances
  (iii) Multiple strongly correlated instance.
  (iv) Strongly correlated and bounded strongly correlated
  (v) Spanner strongly correlated instances
  (vi) Profit ceiling type of instances

- The problem types considered are ordered in decreasing order of gap % in computation time as follows

  (i) Profit ceiling instances
  (ii) Spanner uncorrelated instances
  (iii) Spanner strongly correlated instances.
  (iv) Uncorrelated instances.
  (v) Strongly correlated, bounded strongly correlated and Multiple strongly correlated.

- Thus, QIEA-PSA provides good solutions in lesser time even for those problem types for which QIEA does not provide good solutions. For those types for which QIEA provides good solutions, QIEA-PSA provides similar or better solutions in much lesser time.
- Time taken to reach best solution is almost same as time taken till termination in case of QIEA but it is much less in case of QIEA-PSA.
- FES taken to compute the solutions is quite less in case of QIEA-PSA as compared to QIEA. Moreover each FES in QIEA-PSA take lesser time as compared to QIEA as QIEA-PSA evaluate instances of reduced size after each iteration.
- Based on range of time taken on an average by QIEA-PSA to compute best solution for problems of all sizes (Fig. 13) and time taken by QIEA-PSA to solve instances of size 15,000, a difficulty ranking of problem types is presented in Table 19. The three problem types viz., strongly correlated, bounded strongly correlated, and multiple strongly correlated are ranked as most difficult.

Table 21 Comparing average profit values using MBPSO and QIEA-PSA for set-I of KP instances

Example	MBPSO	QIEA-PSA	Example	MBPSO	QIEA-PSA
ks_8a	3,924,400	3,924,400	ks_16d	9,337,915.6	9,343,887.8
ks_8b	3,813,669	3,813,669	ks_16e	7,764,131.8	7,755,224.2
ks_8c	3,347,452	3,347,452	ks_20a	10,720,314	10,727,049
ks_8d	4,187,707	4,187,707	ks_20b	9,805,480.5	9,818,261
ks_8e	4,954,571.7	4,955,555	ks_20c	10,710,947	10,712,473
ks_12a	5,688,552.4	5,688,757.3	ks_20d	8,923,712.2	8,928,880.8
ks_12b	6,493,130.6	6,498,597	ks_20e	9,355,930.4	9,357,767
ks_12c	5,170,493.3	5,170,626	ks_24a	13,532,060	13,549,094
ks_12d	6,992,144.3	6,992,404	ks_24b	12,223,443	12,233,713
ks_12e	5,337,472	5,337,472	ks_24c	12,443,349	12,448,780
ks_16a	7,843,073.3	7,850,983	ks_24d	11,803,712	11,813,578
ks_16b	9,350,353.4	9,352,998	ks_24e	13,932,526	13,940,099
ks_16c	9,144,118.4	9,151,147			

The profit ceiling is ranked easiest and remaining problems are ranked as average on difficulty level. Similar observations were also made in Sect. 5.1 from graphs of Fig. 7.

- Strongly correlated, bounded strongly correlated, and multiple strongly correlated are found equally harder in both, the QIEA and the QIEA PSA. All of them showed improvement in profit of around 10 % with reduction in time of almost 52 %. Let's refer them as the trio in the following discussion.

An analysis of the above observation yields the following points.

- For Un-correlated and Spanner Uncorrelated **both gap % (time and profit) are high**. It means they are very difficult for QIEA but very easy for QIEA-PSA. That is because searching for good solution of these problems is directionless in simple QIEA where as the heuristic provides better focus to the search process in QIEA-PSA.
- Spanner Strongly Correlated showed profit increment of around 10 % which is obtained faster in QIEA-PSA. Therefore, they are easier than the trio. The reason is that many elements in these are of similar weights and profits and so the QIEA search and local search method finds good solutions in such a solution space faster. Thus this problem is a bit easier in QIEA than the trio while too much easier in QIEA-PSA than the trio.
- For profit ceiling the gap % in profit is not high but that in time is quite high i.e. they are easier than the trio. Here, values of profit and weight for each item are similar. Many items have same profit but slightly different weights. So finding good solution in such a search space is easier. Further the greedy heuristic makes the search easier. These problems are very easy for QIEA but still easier for QIEA-PSA.

5.3 Benchmark instances in Boyer et al. [35]

The QIEA-PSA is used to solve the instances [47] used by Boyer et al. [35] to report the performance of their parallel implementation of exact dynamic programming based algorithm. Table 20 illustrates the comparison of average time obtained for 10 instances using serial implementation of QIEA-PSA with that obtained using parallel implementation of Boyer et al. Average gap % is calculated between profit obtained using QIEA-PSA and the optimal value for an instance as ((Optimal Profit − QIEA-PSA Profit) / Optimal Profit) *100. It is clear that QIEA-PSA gives nearly optimal values in much lesser computation times than the parallel implementation of Boyer. All constants of QIEA-PSA are set to values as mentioned in the beginning of Sect. 5.

5.4 Benchmark instances in Bansal and Deep [40]

Bansal and Deep [40] solved the benchmark KP instances using their Modified Binary Particle Swarm Optimization (MBPSO). The population size for these instances in QIEA-PSA is set to 4000. Two sets of instances, first containing 25 instances is taken from http://www.math.mtu.edu/_kreher/cages/Data.html and the second containing 6 instances is obtained directly from the authors. Table 21 and 22 presents the comparison of results obtained using MBPSO with QIEA-PSA for set I and set II of KP Instances.

The values of gaps in weight of selected items from maximum capacity and the values of maximum profit observed in QIEA-PSA are same as in MBPSO for set-I of KP instances, so comparison is presented only on the basis of average of profit obtained over 100 runs. The QIEA-PSA provides bet-

Table 22 Comparison between MBPSO and QIEA-PSA based on results for set-II of KP instances

Example	Size	Optimal	Algorithm	SR	AFE	AE	LE	Std dev
1	10	295	MBPSO	100	543	0	0	0
			QIEA-PSA	100	1136.32	0	0	0
2	20	1024	MBPSO	100	2952	0	0	0
			QIEA-PSA	100	511.35	0	0	0
3	50	3112	MBPSO	66	62,212	0.68	0	1.4274
			QIEA-PSA	100	1	−2	−2	0
4	100	2,683,223	MBPSO	50	241,805	284.03	0	325.2135
			QIEA-PSA	0	279,029.3	5428.28	1760	955.704
5	200	5,180,258	MBPSO	0	600,000	872.74	25	432.8804
			QIEA-PSA	0	543,228.3	1276.41	189	442.6333
6	500	1,359,213	MBPSO	0	1,500,000	1248.96	586	275.2432
			QIEA-PSA	100	114,041.7	0	0	0

Instance 3 we received from authors is possibly incorrect since initialized profit value in QIEA-PSA itself is more than given optimal

ter profits on an average over 100 runs than MBPSO for all instances except instance ks_16e.

For set-II of instances the optimal profit is known. Table 22 present a comparison on the basis of SR (success rate), AFE (average function evaluations), AE (average error), LE (least error) and StdDev (standard deviation in profit) over 100 runs for each instance. The QIEA-PSA performs significantly better on instances 2 and 6. The performance of two is competitive for instances 1 and 5. The instance 3 obtained from authors seems to have some error since the value obtained in QIEA-PSA for initialization of best solution itself is better than the optimal value mentioned.

Thus, QIEA-PSA performs better than MBPSO on the basis of quality of solutions provided for simple knapsack problems.

6 Conclusions and future work

An improved quantum inspired evolutionary algorithm, dubbed QIEA-PSA, is presented. The modifications introduced here are as follows, initializing and repairing the collapsed qubit individuals based on information provided by heuristic for the instance, reducing the problem size and re-initialization of population of local best solutions for each new generation.

The results for simple QIEA and QIEA-PSA are presented for very large size (290,000 items) selected difficult KP instances.

A comparison presents the effects of various changes done in the algorithm for a sample of instances. A detailed analysis of the impact of various modifications done on QIEA is presented.

A comparison with parallel implementation of a deterministic algorithm shows that solutions provided by QIEA-PSA are very close to optimal.

Another comparison of QIEA-PSA is made with a modified binary particle swarm optimization algorithm recently applied on benchmark KP instances.

To summarize, the following observations result from the experiments:

- The presented QIEA-PSA provides much better results for difficult KP instances than the QIEA both in terms of computing time and quality of solution.
- Greedy Initialization of qubits and best solution when clubbed with size reduction results in more rapid convergence of qubits to their final values.
- The size of problem gets reduced to an insignificantly small portion of original problem by the 10th iteration.
- These improvements together imply that QIEA-PSA gives much better solutions than QIEA in much lesser computation time.
- The time taken by the reported implementation of QIEA-PSA grows almost linearly with problem size and so is able to handle problems that are much larger than those reported in the literature and yet give close to optimal values.
- QIEA-PSA provides close to optimal solutions in much lesser time than even a recent parallel implementation of an exact algorithm by Boyer et al. [35].
- The population size can be increased to solve more difficult instances.
- QIEA-PSA outperforms a modified binary particle swarm optimization algorithm of Bansal and Deep [40] recently applied on benchmark KP instances.

Designing the effective QIEA using the similar modifications for other more complex problems can be considered for further work. The work can also be done such that the algorithm adapts to the appropriate population size automatically depending on the difficulty in solving the instance.

Acknowledgments 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".

References

1. Kellerer H, Pferschy U, Pisinger D (2004) Knapsack problems. Springer, Berin
2. Horowitz E, Sahani S (1974) Computing partitions with applications to the knapsack problem. J ACM 21:277–292
3. Fayard D, Plateau G (1975) Resolution of the 0–1 knapsack problem. Comparison of methods. Math Program 8:272–307
4. Nauss RM (1976) An efficient algorithm for the 0–1 knapsack problem. Manage Sci 23:27–31
5. Martello S, Toth P (1977) An upper bound for the zero-one knapsack problemand a branch and bound algorithm. Eur J Oper Res 1:169–175
6. Bradley GH (1971) Transformation of integer programs to Knapsack problems. Discrete Math 1(1):29–45
7. Karp R (1972) Reducibility among combinatorial problems. Technical Report 3. University of California, Berkeley
8. Narayanan A, Moore M (1996) Quantum-inspired genetic algorithms. In: Proc CEC, pp 61–66
9. Han K, Kim J (2000) Genetic quantum algorithm and its application to combinatorial optimization problem. In: Proc. CEC, pp 1354–1360
10. Han KH, Kim JH (2002) Quantum-Inspired Evolutionary Algorithm for a class of combinatorial optimization. IEEE Trans Evol Comput 6(6):580–593
11. Han KH (2006) On the analysis of the Quantum-inspired Evolutionary Algorithm with a single individual. In: IEEE congress on evolutionary computation, Vancouver, Canada
12. Platel MD, Schliebs S, Kasabov N (2009) Quantum-Inspired Evolutionary Algorithm: a multimodel EDA. IEEE Trans Evol Comput 13(6):1218–1232
13. Zhang G (2011) Quantum-inspired evolutionary algorithms: a survey and empirical study. J Heurist 17(3):303–351
14. Zhang H, Zhang G, Rong H, Cheng J (2010) Comparisons of quantum rotation gates in Quantum-Inspired Evolutionary Algorithms. In: Sixth international conference on natural computation (ICNC 2010), Hiroshima
15. Yang SY, Wang M, Jiao LC (2004) A genetic algorithm based on quantum chromosome. In: Proc. ICSP, pp 1622–1625
16. Yang SY, Wang M, Jiao LC (2004) A novel quantum evolutionary algorithm and its application. In: Proc CEC, pp 820–826
17. Patvardhan C, Prakash P, Srivastav A (2009) A novel quantum-inspired evolutionary algorithm for the quadratic knapsack problem. In: Proceedings of the international conference on operationas research applications in engineering and management, May 2009; Tiruchirapalli, India, pp 2061–2064
18. Zhang GX, Li N, Jin WD, Hu LZ (2006) Novel quantum algorithm and its applications. Front Electr Electron Eng Cina 1(1):31–26
19. Li Y, Zhang YN, Zhao RC, Jiao LC (2004) The immune quantum-inspired evolutionary algorithm. In: IEEE ICSMC, pp 3301–3305
20. Li Y, Zhang Y, Cheng Y, Jiang X, Zhao R (2005) A novel immune quantum-inspired genetic algorithm. In: Lecture notes in computer science, pp 215–218
21. Wang Y, Feng XY, Huang YX, Pu DB, Zhou WG, Liang YC, Zhou CG (2007) A novel quantum swarm evolutionary algorithm and its applications. Neurocomputing 70(4–6):633–640
22. Wang Y, Feng XY, Huang YX, Zhou WG, Liang YC, Zhou CG (2005) A novel quantum swarm evolutionary algorithm for solving 0–1 knapsack problem. In: Lecture notes in computer science, pp 698–704
23. Zhang GX, Gheorghe M, Wu CZ (2008) A quantum-inspired evolutionary algorithm based on p systems for knapsack problem. Fund Inf 87(1):93–116
24. Yarlagadda P, Kim YH (2012) An Improved Quantum-Inspired Evolutionary Algorithm based on P systems with a dynamic membrane structure for Knapsack problems. Appl Mech Mater 239–240:1528–1531
25. Mani A, Patvardhan C (2010) A hybrid quantum evolutionary algorithm for solving engineering optimization problems. Int J Hybrid Intell Syst 7:225–235
26. Mani A, Patvardhan C (2010) Solving ceramic grinding optimization problem by adaptive quantum evolutionary algorithm. In: Proceedings of the international conference on intelligent systems, modelling and simulation, January 2010, Liverpool, United Kingdom
27. Han K, Kim J (2004) Quantum-inspired evolutionary algorithms with a new termination criterion, h-epsilon gate, and two-phase scheme. IEEE Trans Evol Comput 8(2):156–169
28. Han K, Kim J (2003) On setting the parameters of quantum-inspired evolutionary algorithm for practical application. In: Proc. CEC, pp 178–184
29. Han K, Park K, Lee C, Kim J (2001) Parallel quantum-inspired genetic algorithm for combinatorial optimization prblem. In: Proc. CEC, pp 1422–1429
30. Zhang R, Gao H (2007) Improved quantum evolutionary algorithm for combinatorial optimization problem. In: Proc. ICMLC, pp 3501–3505
31. Platel MD, Schliebs S, Kasabov N (2007) A versatile quantum-inspired evolutionary algorithm. In: Proc. CEC, pp 423–430
32. Kim Y, Kim JH, Han KH (2006) Quantum-inspired multiobjective evolutionary algorithm for multiobjective 0/1 knapsack problems. In: Proc. CEC, pp 2601–2606
33. Patvardhan C, Narayan A, Srivastav A (2007) Enhanced Quantum Evolutionary Algorithms for difficult Knapsack problems. In: PReMI'07 Proceedings of the 2nd international conference on Pattern recognition and machine intelligence, pp 252–260
34. Nowotniak R, Kucharski J (2012) GPU-based tuning of quantum-inspired genetic algorithm for a combinatorial optimization problem. Bull Polish Acad Sci Tech Sci 60(2):323–330
35. Boyer V, Baz DE, Elkihel M (2012) Solving knapsack problems on GPU. Comput Oper Res 39:42–47
36. Reilly CH (2009) Synthetic optimization problem generation: show us the correlations!. INFORMS J Comput 21(3):458–467
37. Martello S, Pisinger D, Paolo T (2000) New trends in exact algorithms for the 0–1 knapsack problem. Eur J Oper Res 123(2):325–332
38. Pisinger D (2005) Where are the hard Knapsack problems. Technical Report. Comput Oper Res 32(5):2271–2284
39. Martello S, Pisinger D, Toth P (1999) Dynamic programming and strong bounds for the 0–1 Knapsack problem. Manage Sci 45:414–424
40. Bansal JC, Deep K (2012) A modified binary particle swarm optimization for Knapsack problems. Appl Math Comput 218:11042–11061
41. Martello S, Toth P (1990) Knapsack problems: algorithms and computer implementations. Wiley, Chichester
42. 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, Hangzhou, pp 203–206
43. Tayarani-N MH, Akbarzadeh-T MR (2008) A sinusoid size ring structure quantum evolutionary algorithm. In: IEEE conference on cybernetics and intelligent systems, pp 1165–1170
44. Mahdabi P, Jalili S, Abadi M (2008) A multi-start quantum-inspired evolutionary algorithm for solving combinatorial optimization problems. In: (GECCO '08) Proceedings of the 10th annual conference on genetic and evolutionary computation, pp 613–614
45. 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
46. Pisinger D (2012) David Pisinger's optimization codes. [Internet]. [cited 2012 November 2]. http://www.diku.dk/pisinger/codes.html
47. Boyer V, Baz D, Elkihel M. Knapsack Problems. [Internet]. http://www.laas.fr/CDA-EN/45-31329-Knapsack-problems.php