Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2014, Article ID 976202, 22 pages
http://dx.doi.org/10.1155/2014/976202

Research Article

Effect of Population Structures on Quantum-Inspired Evolutionary Algorithm

Nija Mani,[1] Gursaran Srivastava,[1] A. K. Sinha,[1] and Ashish Mani[2]

[1]*Department of Mathematics, Dayalbagh Educational Institute, Dayalbagh, Agra 282005, India*
[2]*USIC, Dayalbagh Educational Institute, Dayalbagh, Agra 282005, India*

Correspondence should be addressed to Ashish Mani; mani.ashish@gmail.com

Received 4 August 2014; Revised 21 November 2014; Accepted 22 November 2014; Published 24 December 2014

Academic Editor: Zhang Yi

Quantum-inspired evolutionary algorithm (QEA) has been designed by integrating some quantum mechanical principles in the framework of evolutionary algorithms. They have been successfully employed as a computational technique in solving difficult optimization problems. It is well known that QEAs provide better balance between exploration and exploitation as compared to the conventional evolutionary algorithms. The population in QEA is evolved by variation operators, which move the Q-bit towards an attractor. A modification for improving the performance of QEA was proposed by changing the selection of attractors, namely, versatile QEA. The improvement attained by versatile QEA over QEA indicates the impact of population structure on the performance of QEA and motivates further investigation into employing fine-grained model. The QEA with fine-grained population model (FQEA) is similar to QEA with the exception that every individual is located in a unique position on a two-dimensional toroidal grid and has four neighbors amongst which it selects its attractor. Further, FQEA does not use migrations, which is employed by QEAs. This paper empirically investigates the effect of the three different population structures on the performance of QEA by solving well-known discrete benchmark optimization problems.

1. Introduction

Evolutionary algorithms (EAs) represent a class of computational techniques, which draw inspiration from nature [1] and are loosely based on the Darwinian principle of "survival of the fittest" [2–4]. EAs have been successfully applied in solving wide variety of real life difficult optimization problems (i.e., problems which do not have efficient deterministic algorithms for solving them, yet known) and where near optimal solutions are acceptable (as EAs do not guarantee finding optimal solutions). Moreover, EAs are not limited by the requirements of domain specific information as in the case of traditional calculus based optimization techniques [5]. EAs, typically, maintain a population of candidate solutions, which compete for survival from one generation to the next, and, with the generation of new solutions by employing the variation operators like crossover, mutation, rotation gate, and so forth, the population gradually evolves to contain the optimal or near optimal solutions. EAs are popular due to their simplicity and ease of implementation. However, EAs suffer from convergence issues like stagnation, slow convergence, and premature convergence [6].

Efforts have been made by researchers to overcome the convergence issues by establishing a better balance between exploitation and exploration. Quantum-inspired evolutionary algorithm (QEA) [7, 8] provides a better balance between exploration and exploitation during the evolutionary search process by using probabilistic Q-bit. QEAs have performed better than classical EAs on many complex problems [9–17].

Some investigations have also been made in using structured populations to improve the performance of EAs [18]. The structure of a population is classified as follows: panmixia, coarse-grained, and fine-grained models. The QEA described in [7, 19] has employed a population structured as coarse-grained model. The performance of this algorithm has been improved by changing the global update strategy in [8] and has been named versatile QEA (vQEA). The modification to convert QEA into vQEA can be also viewed as equivalent to changing the population structure from coarse-grained model in QEA to panmictic in vQEA. The improvement attained by vQEA over QEA indicates the impact of

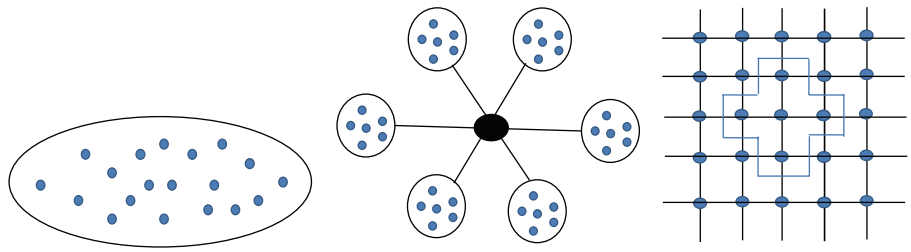


Figure 1: Population structures.

population structure on the performance of QEA. This also motivates investigation for employing fine-grained model in the population structure of QEA. This paper empirically evaluates the effect of population structures in QEA by solving some instances of well-known benchmark problems like COUNTSAT, *P*-PEAKS, and 0-1 knapsack problems. The paper is further organized as follows. Section 2 briefly discusses population structures. QEA, vQEA, and FQEA are presented and their population structure is established in Section 3. Results and analysis are presented in Section 4. Section 5 draws conclusions and shows some directions for further work.

2. Population Structures

The population structures in EAs can be divided into two broad categories, namely, unstructured and structured [20], and are shown in Figure 1. The unstructured population model has been widely used in EAs, where a single population of individuals is evolved by employing variation operators. The advantage of unstructured population is its conceptual and implementation simplicity. The dissemination of information regarding the best individual is quickest as all the individuals are connected with all the other individuals. It works fine for EAs in which diversity can be maintained by suitably designed variation operators for a given problem. However, if the search space is highly deceptive and multimodal, the panmictic population may not be the most effective model. Structured population models are viable alternatives to the unstructured model. They have been primarily developed during efforts to parallelize EAs for running on multiprocessors hardware [2]. However, they have also been used with simple EAs (run on monoprocessor hardware) in place of panmictic population and have been found to provide better sampling of search space along with consequent improvement in empirical performance [21].

The structured models can be divided into two main groups, namely, coarse-grained and fine-grained models. The coarse-grained model is also known as distributed model and island model. It has multiple panmictic populations, which evolve in parallel and communicate with each other periodically, often exchanging or updating individuals, depending on the specific strategy. The advantage of island model is that it encourages niching and also allows slow dissemination of information across the structured subpopulation evolving in parallel. Thus, it maintains diversity in overall population to avoid premature convergence. However, it is known to be relatively slow in converging to optimal solution.

Fine-grained model is also known as cellular model and diffusion model. A unique coordinate is assigned to every individual of the single population in some space, which is typically a grid of some dimensionality with fixed or cyclic boundary. Individuals can only interact within a neighborhood, defined by neighborhood topology, through variation operators. The advantage of cellular model is slow diffusion of information through overlapped small neighborhoods which helps in exploring the search space by maintaining better diversity than panmictic model. This in turn helps in avoiding premature convergence [22]. Moreover, it is slower than a corresponding panmictic model in a given EA but is faster than a coarse-grained model. It has less communication overhead than a coarse-grained model as it maintains a single structured population rather than multiple subpopulations evolving in parallel. Further, it has been shown that a cellular model is more effective in complex optimization tasks as compared to the other models [23, 24].

3. Quantum-Inspired Evolutionary Algorithms

Quantum-inspired proposals are subset of a much larger attempt to apply quantum models in information processing, which is also referred to as *quantum interaction* [25]. The potential advantages of parallelism offered by quantum computing [26] through superposition of basis states in qubit registers and simultaneous evaluation of all possible represented states have led to the development of approaches that suggest ways to integrate aspects of quantum computing with evolutionary computation [25]. Most types of hybridization have focused on designing algorithms that would run on conventional computers and not on quantum computers and are most appropriately classified as "quantum-inspired." The first attempt was made by Narayanan and Moore [27] to use quantum interpretation for designing a quantum-inspired genetic algorithm. A number of other types of hybridization have also been proposed, of which the most popular is the proposal made by Han and Kim [7], which uses a Q-bit as the smallest unit of information and a Q-bit individual as a string of Q-bits rather than binary, numeric, or

symbolic representations. Results of experiments show that QEA performs well, even with a small population, without premature convergence as compared to the conventional genetic algorithms. Experimental studies have also been reported by Han and Kim to identify suitable parameter settings for the algorithm [19]. Platel et al. [8] have shown the weaknesses of QEA and proposed a new algorithm, called versatile quantum-inspired evolutionary algorithm (vQEA). They claim that vQEA is better than the QEA [7] as it guides the search towards the best solution found in the last iteration, which facilitates smoother and more efficient exploration. This claim is supported by experimental evaluations.

A qubit is the smallest information element in a quantum computer and is quantum analog of classical bit. The classical bit can be either in state "zero" or in state "one" whereas a quantum bit can be in a superposition of basis states in the quantum system. It is represented by a vector in Hilbert space with $|0\rangle$ and $|1\rangle$ being the basis states. The qubit can be represented by vector $|\psi\rangle$, which is given by

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{1}$$

where $|\alpha|^2$ and $|\beta|^2$ are the probability amplitudes of qubit to be in state $|0\rangle$ and $|1\rangle$, respectively, and should satisfy the following condition:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2}$$

The QEA proposed in [7, 19] primarily hybridizes the superposition and measurement principles of quantum mechanics in evolutionary computing framework by implementing qubit as Q-bit, which is essentially a probabilistic bit, and stores α and β values. A Q-bit string acts as the genotype of an individual and the binary bit string formed by collapsing the Q-bit forms the phenotype of the individual. A Q-bit is modified by using quantum gates or operators, which are also unitary in nature, as restricted by the postulates of linear quantum mechanics [26]. The quantum gates are implemented in QEA as unitary matrix [7]. A quantum gate known as rotation gate has been employed in [19] and it updates a Q-bit in the following manner:

$$\begin{bmatrix} \alpha_i^{t+1} \\ \beta_i^{t+1} \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i^{t} \\ \beta_i^{t} \end{bmatrix}, \tag{3}$$

where α_i^{t+1} and β_i^{t+1} denote probabilities of ith Q-bit in $(t+1)$th iteration. $\Delta\theta_i$ is the angle of rotation. It acts as the main variation operator that rotates Q-bit strings to obtain good candidate solutions for the next generation. It also requires an attractor [8] towards which the Q-bit will be rotated. It further takes into account the relative current fitness level of the individual and the attractor and also their binary bit values for determining the magnitude and direction of rotation. The selection of an attractor is determined by the population model employed in a QEA. A close scrutiny of the architecture of QEAs designed in [7, 8, 19] reveal that attractors are the only mechanism available for interaction between the individuals.

The QEA in [7, 19] divides the population into local groups and implements two types of migrations, namely, local and global. In local migration, the best solution within the group is used as the attractor, whereas, in case of global migration, the global best solution is used as the attractor. The migration periods and size of local group are design parameters and have to be chosen appropriately for the problem being solved.

The quantum-inspired evolutionary algorithm is shown in Algorithm 1 [7, 8].

In step (a), `Q(t)` containing Q-bit strings for all the individuals are initialized randomly. In step (b), the binary solutions in $P(0)$ are constructed by measuring the states of $Q(0)$. The process of measuring or collapsing Q-bit is performed by generating a random number between 0 and 1 and comparing it with $|\alpha|^2$. If the random number is less than $|\alpha|^2$, then the Q-bit collapses to 0 or else to 1 and this value is assigned to the corresponding binary bit. In step (c), each binary solution is evaluated to give a measure of its fitness. In step (d), the initial solutions are then stored for each individual into $B(0)$. In step (e), the initial best solution for each group, G_j, is then selected and stored into respective $GB_j(0)$. In step (f), the initial global best solution, b, is then selected amongst $GB_j(0)$. In step (g), the attractors are selected for each individual according to the strategy decided by migration criteria. In case of local migration, the group best, $GB_j(t)$, is used as the attractor whereas, in case of global migration, global best, b, is used. In step (h), Q-bit individuals in $Q(t)$ are updated by applying Q-gates by taking into account $AR(t)$ and $P(t)$. The quantum rotation gate has been used as the variation operator. In steps (i) and (j), the binary solutions in $P(t)$ are formed by observing the states of $Q(t-1)$ as in step (c), and each binary solution is evaluated for the fitness value. In step (k), the best solutions among $B(t-1)$ and $P(t)$ are selected and stored into $B(t)$. In step (l), the best solution for each group, G_j, is then selected amongst $GB_j(t-1)$ and $B(t)$ and stored into respective $GB_j(t)$. In step (m), the global best solution, b, is then selected amongst $GB_j(t)$.

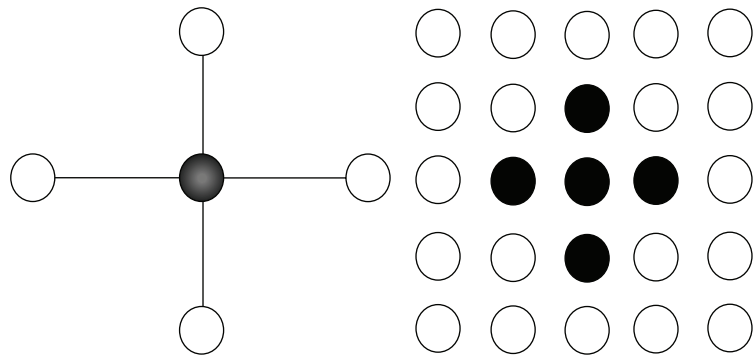
It is suggested that QEA designed in [7] should have population divided equally in 5 groups, where attractors in each group are individuals with best fitness. There is a fixed global migration cycle, at the end of which the attractors are selected as the individual with the best fitness in the entire population. Thus, upon comparing the population structure of QEA in [7, 19] with coarse-grained model, the groups are islands of subpopulation, which interact with each other during global migration that occurs after fixed number of generations. The vQEA [8] does away with the local groups by making global migration in every generation; thus, the population model is now panmictic and the interaction is taking place between every individual in every generation. Thus, the QEA with panmictic population model is referred to as PQEA and the QEA with coarse-grained population structure is referred to as CQEA.

The QEA with fine-grained population model, FQEA, has all the operators and strategies similar to those used in CQEA and PQEA except for the population structure and the neighborhood topology. The fine-grained population model does not have local groups. Every individual is located in a unique position on a two-dimensional toroidal grid as

S_{POP}: Size of the population that is, number of individuals.
S_{GRP}: Size of the Group that is, population is divided in groups.
S_{PB}: Size of the Problem being solved that is, number of variables.
t: Generation Counter.
q_k: kth Q-bit that stores the value of their α and β.
Q_i: ith Quantum Individual comprising of their q_k, where $k = 1, \ldots, S_{PB}$.
$Q(t)$: Quantum Register that comprises of all the Quantum individuals, Q_i, where $i = 1, \ldots, S_{POP}$.
p_k: kth binary bit that stores the value of 0 or 1 formed by collapsing corresponding q_k.
P_i: ith Binary Individual comprising of their p_k, where $k = 1, \ldots, S_{PB}$.
$P(t)$: Binary Register that comprises of all the Binary individuals, P_i, where $i = 1, \ldots, S_{POP}$.
$B(t)$: stores the best solution of all the Binary individuals, P_i, where $i = 1, \ldots, S_{POP}$.
$GB_j(t)$: stores the Best solution of Group, G_j, in the current (tth) generation.
AR(t): Attractor Register that stores the attractor individual for every Q_i, where $i = 1, \ldots, S_{POP}$.
b: current Global Best Solution.

```
begin
    t = 0; assign S_POP, S_GRP, S_PB;
(a) initialize Q(t);
(b) make P(t) by observing the states of Q(t);
(c) evaluate P(t);
(d) store the best solutions among P(t) into B(t);
(e) stores the best solution in each Group, G_j, into respective GB_j(t), j = 1,...,S_POP/S_GRP;
(f) store the best solution b amongst GB_j(t);
    while (termination condition is not met) do
    begin
      t = t + 1;
(g)   select AR(t) according to migration condition;
(h)   update Q(t−1) according to P(t−1) and AR(t) using Q-gates;
(i)   make P(t) by observing the states of Q(t);
(j)   evaluate P(t);
(k)   store the best solutions among P(t) and B(t−1) into B(t);
(l)   store the best solution in each Group, G_j, and GB_j(t−1) into GB_j(t) respectively,
      j = 1,...,S_POP/S_GRP;
(m)   store the best solution b amongst GB_j(t);
    end
end
```

Algorithm 1



(a) Von-Neumann topology

(b) Von-Neumann topology on two-dimensional grid

Figure 2: Details of fine-grained population structure.

shown in Figure 2, which is the most common topology in fine-grained model. The size of the grid is "A cross B," where "A" is the number of rows and "B" is the number of columns. The neighborhood on the grid is defined by Von-Neumann topology, which has five individuals, that is, the current individual and its immediate north, east, west, and south neighbors. Thus, it is also called NEWS or linear 5 (L5) neighborhood topology. The neighborhood is kept static in the current work so it is computed only once during a single run.

S_{POP}: Size of the population that is, number of individuals.
S_{PB}: Size of the Problem being solved that is, number of variables.
A: Number of Rows
B: Number of Columns
t: Generation Counter.
q_k: kth Q-bit that stores the value of their α and β.
Q_i: ith Quantum Individual comprising of their q_k, where $k = 1, \ldots, S_{PB}$.
$Q(t)$: Quantum Register that comprises of all the Quantum individuals, Q_i, where $i = 1, \ldots, S_{POP}$.
p_k: kth binary bit that stores the value of 0 or 1 formed by collapsing corresponding q_k.
P_i: ith Binary Individual comprising of their p_k, where $k = 1, \ldots, S_{PB}$.
$P(t)$: Binary Register that comprises of all the Binary individuals, P_i, where $i = 1, \ldots, S_{POP}$.
$B(t)$: stores the best solution of all the Binary individuals, P_i, where $i = 1, \ldots, S_{POP}$.
AR(t): Attractor Register that stores the attractor individual for every Q_i, where $i = 1, \ldots, S_{POP}$.
b: current Global Best Solution.
NL: List of neighbors of all the individuals, Q_i, where $i = 1, \ldots, S_{POP}$.

```
Begin
    t = 0; assign S_POP, S_PB, A, B;
(a) initialize Q(t);
(b) make P(t) by observing the states of Q(t);
(c) evaluate P(t);
(d) store the best solutions among P(t) into B(t);
(e) compute NL;
(f) store the best solution b amongst B(t);
    while (termination condition is not met) do
    begin
      t = t + 1;
(g)   select AR(t);
(h)   update Q(t−1) according to P(t−1) and AR(t) using Q-gates;
(i)   make P(t) by observing the states of Q(t);
(j)   evaluate P(t);
(k)   store the best solutions among P(t) and B(t−1) into B(t);
(l)   stores the best solution b amongst B(t);
    end
end
```

Algorithm 2

The steps in FQEA are shown in Algorithm 2 [7, 8].

The steps (a) to (d) are the same as those for the QEA described earlier. In step (e), the neighborhood list NL is computed for each individual in the population. In step (f), the initial global best solution, b, is then selected amongst $B(0)$. In step (g), the attractors are selected for each individual by selecting the fittest neighbor from the four neighbors listed in the neighborhood list, NL, of each individual. The steps (h) to (k) are the same as those for the QEA described earlier. In step (l), the global best solution, b, is then selected amongst $B(t)$. Further, there are no local or global migrations as well as local isolated groups in FQEA.

The computation of neighborhood list, NL, is an additional component in FQEA as compared to the QEAs. The neighborhood list, NL, is computed only once during a single run of FQEA, so the overhead involved in computation of the neighborhood is dependent on the population size, S_{POP}, and the number of neighbors of each individual in the population and is independent of the size of the optimization problem being solved and the number of generations executed in a run of FQEA.

The implementation of selection of attractors for all the individuals is different for the three QEAs. It is the simplest and cheapest for PQEA as the global best solution, b, is the attractor for all the individuals. The selection of attractors is dependent on the local group size and the population size in CQEA whereas, in FQEA, it is dependent on the neighborhood size and the population size, so if the local group size and the neighborhood size are equal along with the population size in CQEA and FQEA, then the selection of attractors is equally expensive in both CQEA and FQEA. The rest of the functions has the same implementation in all the QEAs and is equally expensive.

4. Testing, Results, and Analysis

The testing has been performed to evaluate the effect of all the three population models on QEA, namely, coarse-grained QEA (CQEA), panmictic QEA (PQEA), and fine-grained QEA (FQEA) on discrete optimization problems. The testing is performed with equivalent parameter setting for all the three algorithms so that a fair comparison of the impact of three population structures on the performance of QEA can be made statistically. An important point to note here is that the parameters suggested in [7, 8, 17] especially θ_1 to θ_8 have been mostly used as they all use the same variation operator

Table 1: Parameter setting for PQEA, CQEA, and FQEA.

Parameters	PQEA	CQEA	FQEA
Population size	50	50	50
θ_1 to θ_8		$0, 0, 0.01\pi, 0, -0.01\pi, 0, 0, 0$, respectively	
Number of observations	1	1	1
Local group size	N.A.	5	N.A.
Local migration period (generations)	N.A.	1	N.A.
Global migration period (generations)	1	100	N.A.
Toroidal grid size	N.A.	N.A.	5 cross 10
Neighborhood topology	N.A.	N.A.	Von-Neumann
Neighborhood size	N.A.	N.A.	5
Stopping criterion (generations)		Problem specific	

and our main motive has been to determine the effect of different population structures, keeping all the other factors simple and similar.

The testing is performed on thirty-eight instances of well-known diverse problems, namely, COUNTSAT (nine instances), *P*-PEAKS (five instances), and 0-1 knapsack problems (eight instances each for three different profit and weight distributions). COUNTSAT and *P*-PEAKS problem instances have been selected as their optimal values are known; thus, it becomes easier to empirically evaluate the performance of algorithms. 0-1 knapsack problems have been selected as they have several practical applications. The problem instances used in testing of the QEAs are generally large and difficult and are explained in detail in respective Sections 4.1 to 4.3.

The parameters used for all the three algorithms in all the problems are given in Table 1. A population size of fifty and number of observations per Q-bit as one in each generation have been used in all the three algorithms. The value of θ_1 to θ_8 is the same for all the three QEAs. The local group size is five, local migration period is one iteration, and the global migration period is 100 generations for CQEA. The global migration period is one in PQEA. The toroidal grid size of "5 cross 10" with Von-Neumann topology having neighborhood size of five individuals is used in FQEA. The toroidal grid size of "5 cross 10" has been used as the population size is five. The stopping criterion is the maximum number of permissible generations, which is problem specific.

4.1. COUNSAT Problem [18]. It is an instance of the MAXSAT problem. In COUNTSAT, the value of a given solution is the number of satisfied clauses (among all the possible Horn clauses of three variables) by an input composed of n Boolean variables. It is easy to check that the optimum is obtained when the value of all the variables is 1; that is, $s = n$. In this study, nine different instances have been considered with n = 20, 50, 100, 150, 200, 400, 600, 800, and 1000 variables, and, thus, the value of the optimal solution varies from 6860 to 997003000 as given by

$$f_{\text{COUNTSAT}}(s) = s + n \cdot (n-1) \cdot (n-2) - 2 \cdot (n-2) \cdot \binom{s}{2} + 6 \cdot \binom{s}{3}$$

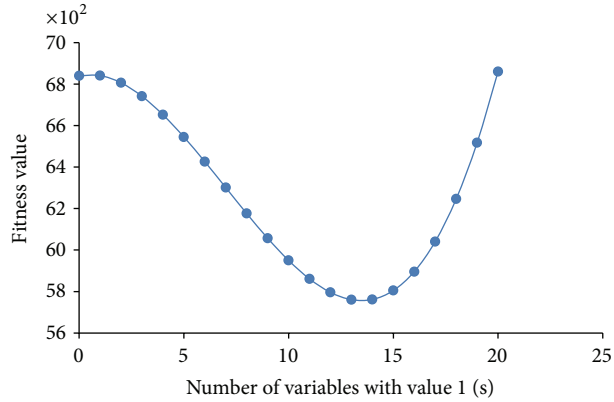


Figure 3: COUNTSAT function with n being 20.

$$\text{For } (n = 20) = s + 6840 - 18 \cdot s \cdot (s-1) + s \cdot (s-1) \cdot (s-2)$$
$$\text{For } (n = 1000) = s + 997002000 - 998 \cdot s \cdot (s-1) + s \cdot (s-1) \cdot (s-2). \quad (4)$$

The COUNTSAT function is extracted from MAXSAT with the objective of being very difficult to be solved by evolutionary algorithms [28]. The variables are randomly assigned values, following a uniform distribution, and so will have approximately $n/2$ ones. Then, the local changes decreasing the number of ones will lead to better results, while local changes increasing the number of ones decrease the fitness as shown in Figures 3 and 4. Hence, it is expected that EAs would quickly converge to all-zero and have difficulties in reaching the all-one string.

The results of testing of all the three algorithms on nine COUNTSAT problem instances have been presented in Table 2. A total of thirty independent runs of each algorithm were executed and the Best, Worst, Average, Median, % success runs, that is, number of runs in which an optimum solution was reached, and standard deviation (Std) of the fitness along with the average number of function evaluations (NFE) were recorded. The maximum number of generations

Table 2: Comparative study between PQEA, CQEA, and FQEA using statistical results on COUNTSAT problem instances.

Problem size (n)	Statistic	Best	Worst	Average	Median	% success runs	Std.	Average NFE
20	PQEA	**6860**	6841	6857	**6860**	83.3	7.2	9091.7
	CQEA	**6860**	**6860**	**6860**	**6860**	**100.0**	**0.0**	1151.7
	FQEA	**6860**	**6860**	**6860**	**6860**	**100.0**	**0.0**	**1138.3**
50	PQEA	**117650**	**117601**	117639	**117650**	76.7	21.1	14256.7
	CQEA	**117650**	**117650**	**117650**	**117650**	**100.0**	**0.0**	4906.7
	FQEA	**117650**	**117650**	**117650**	**117650**	**100.0**	**0.0**	**4488.3**
100	PQEA	**970300**	970201	970270.3	**970300**	70.0	46.1	20268.3
	CQEA	**970300**	**970300**	**970300**	**970300**	**100.0**	**0.0**	**10461.7**
	FQEA	**970300**	970201	970296.7	**970300**	96.7	18.1	10650.0
150	PQEA	**3307950**	3307801	3307890.4	3307950	60.0	74.2	27303.3
	CQEA	**3307950**	3307801	3307925.2	3307950	83.3	56.5	20863.3
	FQEA	**3307950**	**3307950**	**3307950**	**3307950**	**100.0**	**0.0**	**13345.0**
200	PQEA	**7880600**	7880401	7880520.4	**7880600**	60.0	99.2	29723.3
	CQEA	**7880600**	7880401	7880580.1	**7880600**	90.0	60.7	22300.0
	FQEA	**7880600**	**7880600**	**7880600**	**7880600**	**100.0**	**0.0**	**16605.0**
400	PQEA	**63521200**	62847292	63425153	**63521200**	76.7	189932.2	41148.3
	CQEA	**63521200**	63258981	63512459	**63521200**	96.7	47874.4	34483.3
	FQEA	**63521200**	**63521200**	**63521200**	**63521200**	**100.0**	**0.0**	**27968.3**
600	PQEA	**214921800**	209608152	212033464	211953199	10.0	1407346.4	49963.3
	CQEA	**214921800**	209881401	213598979	214385692	36.7	1524388.0	48731.7
	FQEA	**214921800**	**214921800**	**214921800**	**214921800**	**100.0**	**0.0**	**37201.7**
800	PQEA	499089026	488046846	493854763	494709763	0.0	2850315.6	50050.0
	CQEA	508810386	498371881	504276211	504764499	0.0	2502890.2	50050.0
	FQEA	**510082400**	**510082400**	**510082400**	**510082400**	**100.0**	**0.0**	**45186.7**
1000	PQEA	964320421	937999876	951762456	953408640	0.0	5614429.9	50050.0
	CQEA	979662826	948748605	969462614	972362722	0.0	8750510.8	50050.0
	FQEA	**997003000**	**994023961**	**995543257**	**996005997**	**6.7**	**770362.2**	**50010.0**

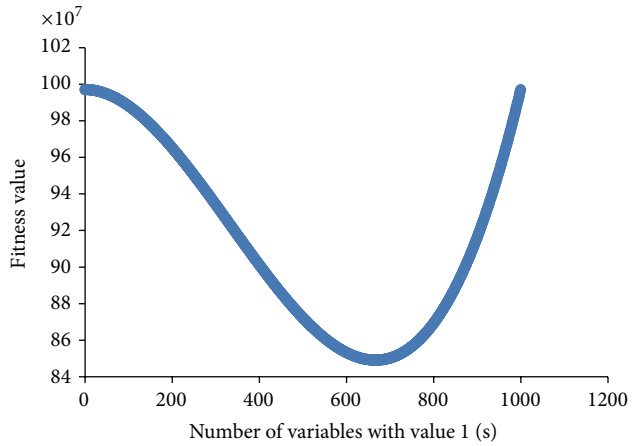


Figure 4: COUNTSAT function with n being 1000.

was one thousand. All the three algorithms were able to reach the global maxima till problem size, n, being 600; however, in problem sizes, n, being 800 and 1000, only FQEA could reach the global optima. The performance of PQEA is inferior to the other two algorithms on all the statistical parameters. The performance of CQEA is as good as FQEA for problem sizes 20 and 50 on all the statistical parameters except for average NFE, which indicates that FQEA is faster than CQEA. CQEA has better success rate than FQEA in the problem size, n, being 100, but FQEA has performed better than CQEA on the remaining six problem instances as it has better success rate.

Figures 5, 6, 7, 8, 9, 10, 11, 12, and 13 show relative convergence rate of the QEAs on the COUNTSAT problem instances. The convergence graphs have been plotted between the number of generations and the objective function values of the three QEAs. The rate of convergence of PQEA is fastest during the early part of the search in all the problem instances. In fact, for small size problem instances, PQEA is the fastest QEA. However, as the problem size increases, the performance of PQEA deteriorates and FQEA, which was slowest on the small size problem instances, emerges as the fastest amongst all the QEAs. CQEA has been faster than FQEA on small size problem instances and has also outperformed PQEAs on large size problem instances.

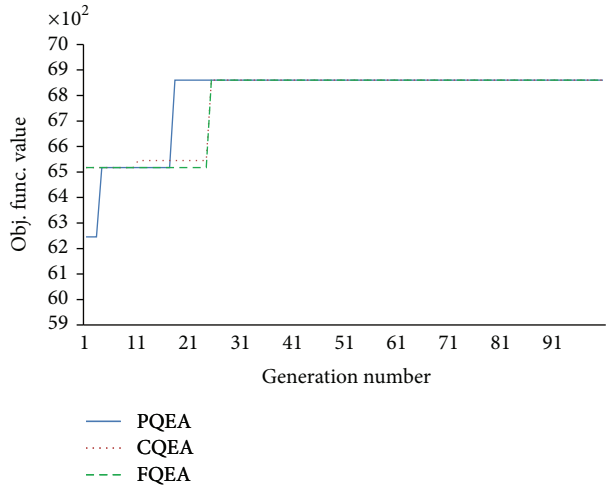


Figure 5: Convergence graph of PQEA, CQEA, and FQEA on COUNTSAT problem size, n, being 20.

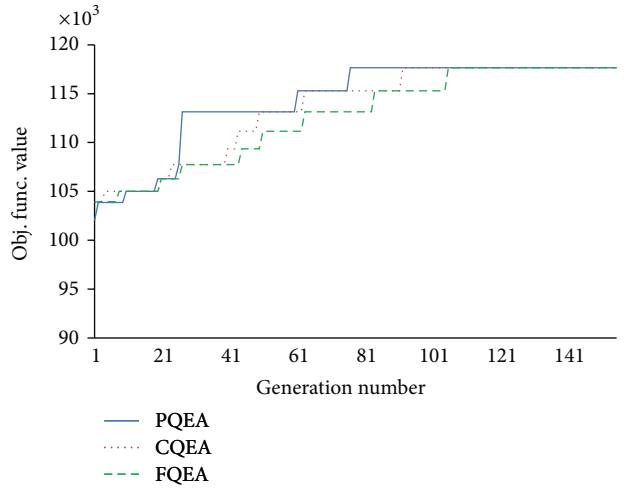


Figure 6: Convergence graph of PQEA, CQEA, and FQEA on COUNTSAT problem size, n, being 50.

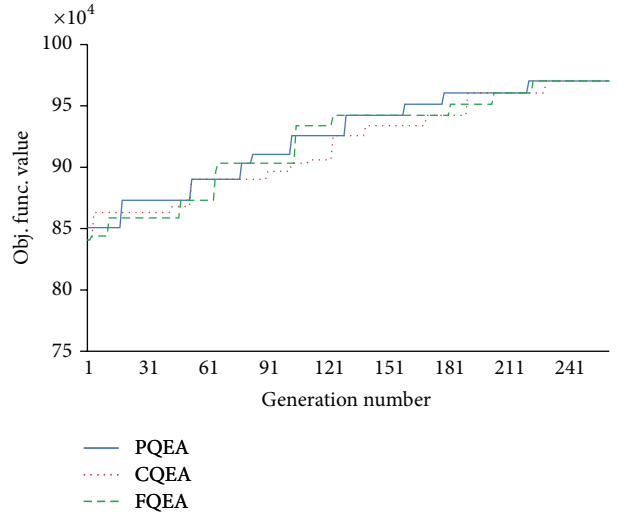


Figure 7: Convergence graph of PQEA, CQEA, and FQEA on COUNTSAT problem size, n, being 100.

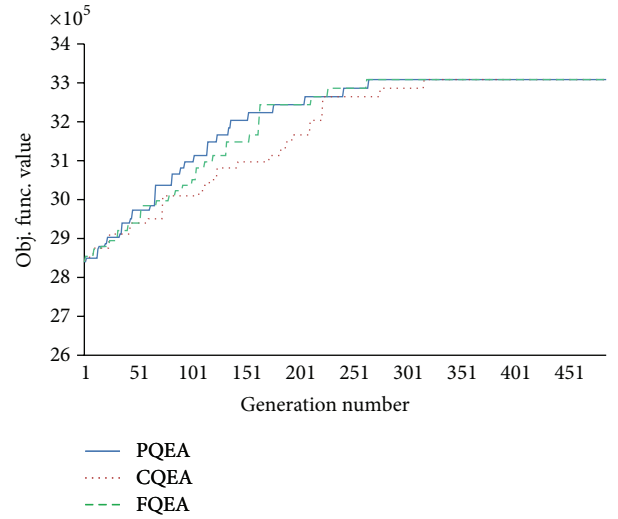


Figure 8: Convergence graph of PQEA, CQEA, and FQEA on COUNTSAT problem size, n, being 150.

The reason for poor performance of FQEA initially on small size problem instances is the slowest dispersion of information as compared to PQEA and CQEA, which, in fact, enables FQEA to explore the solution space more comprehensively before converging to the global optimum. In fact, slow dispersion of information helps FQEA to reach the global optimum in large size problem instances as it does not get trapped in the local optimum. The dispersion of information is quickest in the case of PQEA, which helps it to outperform both CQEA and FQEA, but also causes it to get trapped in the local optimum, especially in large size problem instances. The dispersion of information is slower in CQEA as compared to PQEA so it has found global optimum in more numbers of problem instances. Overall, FQEA has performed better than PQEA and CQEA on all the instances of COUNSAT problems. Therefore, the slow rate of dispersion of information in the fine-grained model had helped FQEA to perform better than the QEAs with the other two population models in the COUNTSAT problem instances.

4.2. P-PEAKS Problem [18, 29]. It is a multimodal problem generator, which can easily create problem instances, which have tunable degree of difficulty. The advantage of using a problem generator is that it removes the opportunity to hand-tune algorithms to a particular problem, thus, allowing a large fairness while comparing the performance of different algorithms or different instances of the same algorithm. It helps in evaluating the algorithms on a large number of random problem instances, so that the predictive power of the results for the problem class as a whole is very high [29].

The idea of P-PEAKS is to generate "P" random N-bit strings that represent the location of P peaks in the search

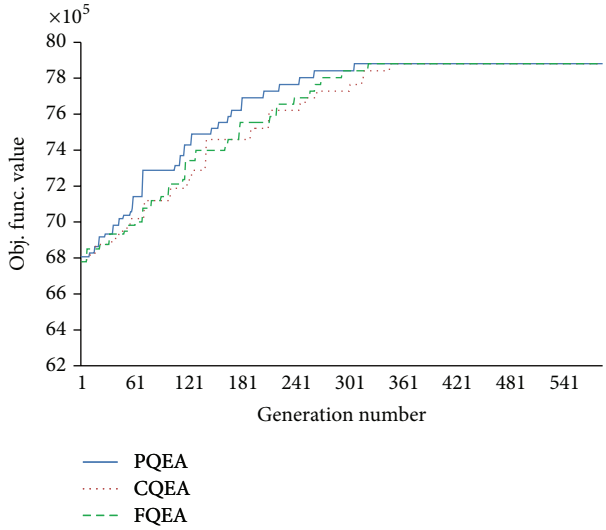


Figure 9: Convergence graph of PQEA, CQEA, and FQEA on COUNTSAT problem size, n, being 200.

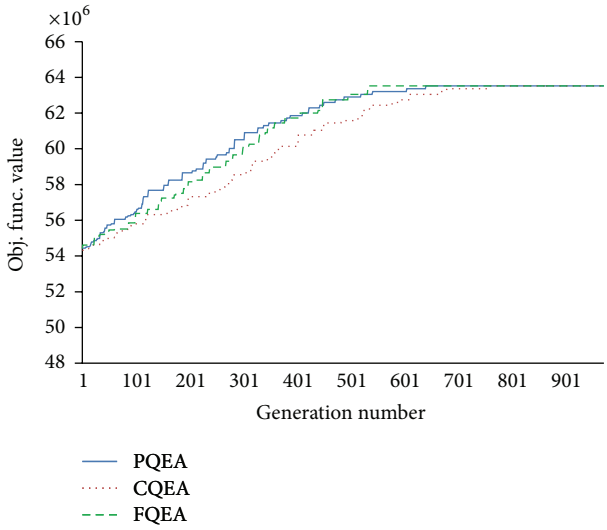


Figure 10: Convergence graph of PQEA, CQEA, and FQEA on COUNTSAT problem size, n, being 400.

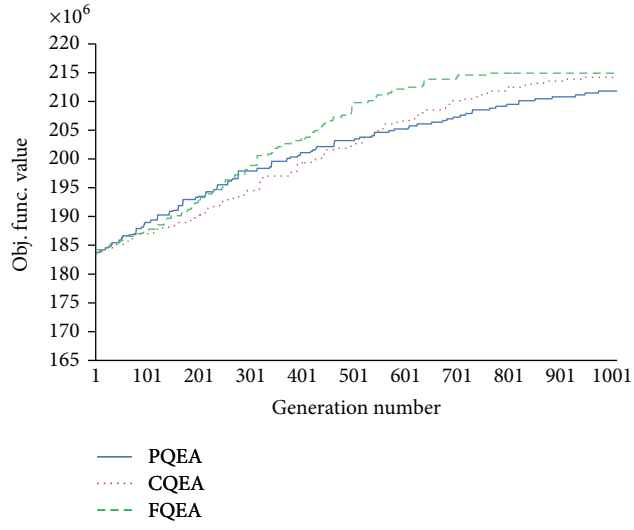


Figure 11: Convergence graph of PQEA, CQEA, and FQEA on COUNTSAT problem size, n, being 600.

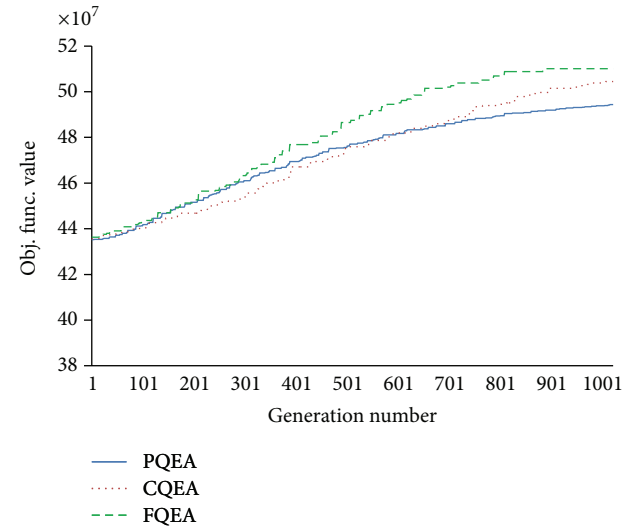


Figure 12: Convergence graph of PQEA, CQEA, and FQEA on COUNTSAT problem size, n, being 800.

space. The fitness value of a string, $\vec{x}$, is the hamming distance between $\vec{x}$ and the closest peak, Peak_i, divided by N (as shown in (5)). Using a higher (or lower) number of peaks, we obtain a problem with more (or less) degree of multimodality. The maximum fitness value for the problem instances is 1.0. Consider

$$f_{P\text{-PEAKS}(\vec{x})} = \frac{1}{N} \max_{1 \le i \le p} \{N - \text{Hamming } D(\vec{x}, \text{Peak}_i)\}. \quad (5)$$

The results of testing of all three algorithms on P-PEAKS problem instances with N being 1000 and number of peaks, P, being 20, 50, 100, 500, and 1000 have been presented in Table 3. A total of thirty independent runs of each algorithm were executed and the Best, Worst, Average, and Median, % success runs, that is, number of runs in which an optimum solution was reached, and standard deviation (Std) of the fitness along with average number of function evaluations (NFE) were recorded. The maximum number of generations was three thousand. PQEA was quickest in the beginning of the search process but could not reach a global optimum even in a single run of any problem instance. CQEA was slow in the beginning but performed better than PQEA. FQEA outperformed all the other QEAs as it was able to reach a global optimum in all the runs of all the problem instances.

Figures 14, 15, 16, 17, and 18 show relative convergence rate of the QEAs for P-PEAKS problem instances. The convergence graphs have been plotted between the number of generations and the objective function values of the three QEAs. The rate of convergence of PQEA is fastest during the early part of the search in all the five problem instances;

Table 3: Comparative study between PQEA, CQEA, and FQEA using statistical results on P-PEAKS problem instances with size N as 1000.

Number of peaks (P)	Algo	Best	Worst	Average	Median	% success runs	Std	Average NFE
20	PQEA	0.982	0.958	0.970	0.971	0.0	0.006	150050
	CQEA	**1.000**	0.977	0.988	0.987	6.7	0.007	149448
	FQEA	**1.000**	**1.000**	**1.000**	**1.000**	**100.0**	**0.000**	**82393**
50	PQEA	0.986	0.956	0.970	0.969	0.0	0.009	150050
	CQEA	0.998	0.956	0.980	0.981	0.0	0.009	150050
	FQEA	**1.000**	**1.000**	**1.000**	**1.000**	**100.0**	**0.000**	**84097**
100	PQEA	0.986	0.952	0.970	0.972	0.0	0.008	150050
	CQEA	0.999	0.971	0.985	0.984	0.0	0.007	150050
	FQEA	**1.000**	**1.000**	**1.000**	**1.000**	**100.0**	**0.000**	**93028**
500	PQEA	0.981	0.954	0.968	0.968	0.0	0.007	150050
	CQEA	0.994	0.966	0.982	0.983	0.0	0.007	150050
	FQEA	**1.000**	**1.000**	**1.000**	**1.000**	**100.0**	**0.000**	**95803**
1000	PQEA	0.977	0.948	0.965	0.966	0.0	0.008	150050
	CQEA	0.994	0.969	0.982	0.984	0.0	0.006	150050
	FQEA	**1.000**	**1.000**	**1.000**	**1.000**	**100.0**	**0.000**	**94495**

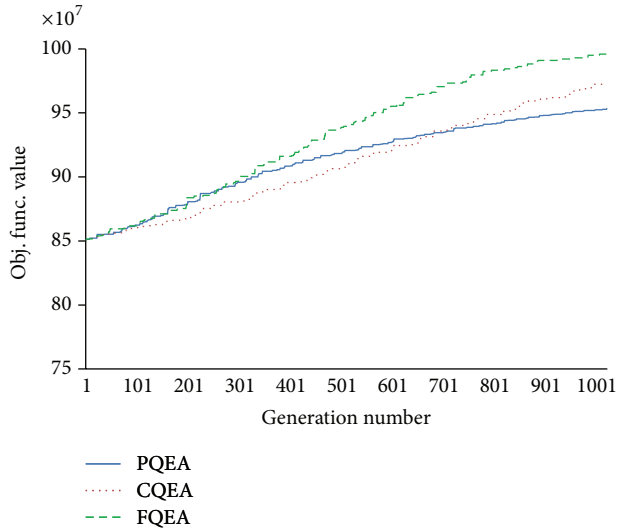


Figure 13: Convergence graph of PQEA, CQEA, and FQEA on COUNTSAT problem size, n, being 1000.

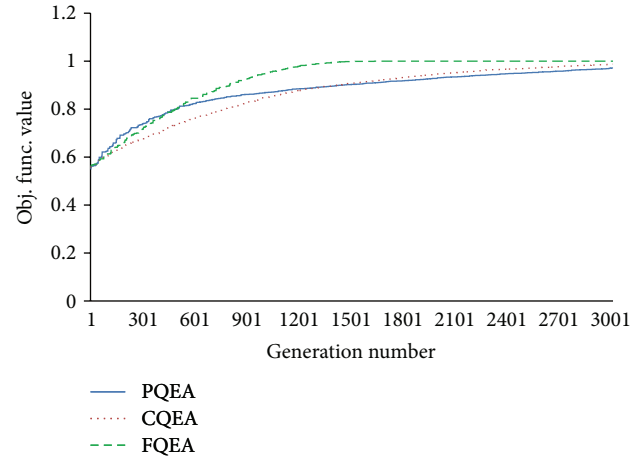


Figure 14: Convergence graph of PQEA, CQEA, and FQEA on P-PEAKS problem of size, N, being 1000 and number of peaks, P, being 20.

however, PQEA could not reach a global optimum even in a single run of any instance. CQEA has been the slowest of the three QEAs but has performed better than PQEA. FQEA is faster than CQEA and has also outperformed PQEA and CQEA on all the five problem instances. The reason for poor performance of FQEA initially is due to the slow dispersion of information as compared to PQEA, which enables it to explore the solution space more comprehensively before converging to a global optimum. In fact, slow dispersion of information helps FQEA to reach a global optimum in large size problem instances as it does not get trapped in a local optimum. The dispersion of information is the quickest in case of PQEA, which helps it to outperform CQEA and FQEA in the initial part of search process but also causes it to get trapped in a local optimum. The dispersion of information is slower in CQEA as compared to PQEA so it has found a global optimum in some runs of a problem instance. Overall, FQEA has performed better than PQEA and CQEA on all the instances of P-PEAKS problems. Therefore, the slow rate of dispersion of information in fine-grained model had helped FQEA to perform better than the other population models in P-PEAKS problem instances.

4.3. 0-1 Knapsack Problem [30]. The 0-1 knapsack problem is a profit maximization problem, in which there are M items of different profit and weight available for selection [30]. The selection is made to maximize the profit while keeping

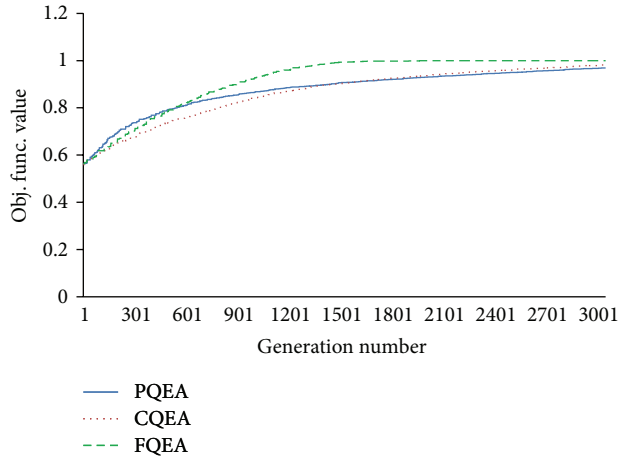


Figure 15: Convergence graph of PQEA, CQEA, and FQEA on P-PEAKS problem of size, N, being 1000 and number of peaks, P, being 50.

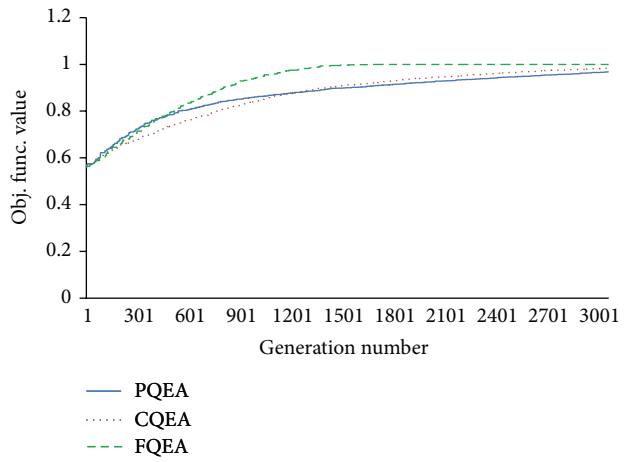


Figure 16: Convergence graph of PQEA, CQEA, and FQEA on P-PEAKS problem of size, N, being 1000 and number of peaks, P, being 100.

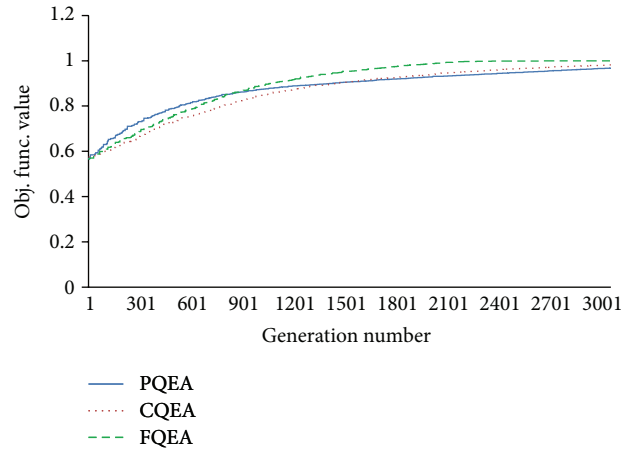


Figure 17: Convergence graph of PQEA, CQEA, and FQEA on P-PEAKS problem of size, N, being 1000 and number of peaks, P, being 500.

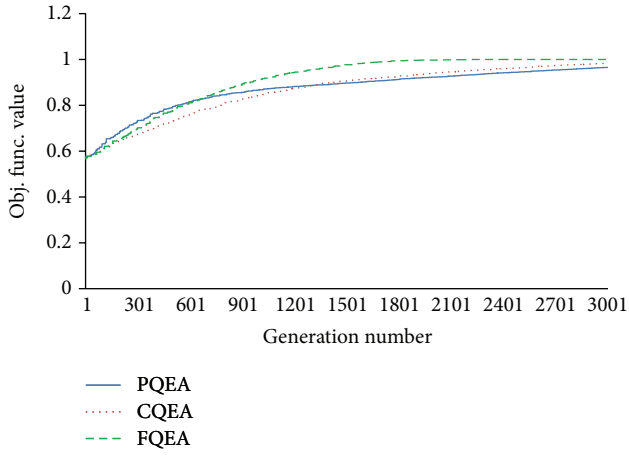


Figure 18: Convergence graph of PQEA, CQEA, and FQEA on P-PEAKS problem of size, N, being 1000 and number of peaks, P, being 1000.

the weight of the selected items below the capacity of the knapsack. It is formulated as follows.

Given a set of M items and a knapsack of capacity C, select a subset of the items to maximize the profit $f(x)$:

$$f(x) = \sum \mathrm{pt}_i * x_i \tag{6}$$

subject to the condition

$$\sum \mathrm{wt}_i * x_i < C, \tag{7}$$

where $x_i = (x_1, \ldots, x_M)$, x_i is 0 or 1, pt_i is the profit of ith item, and wt_i is the weight of ith item. If the ith item is selected for the knapsack, $x_i = 1$; else $x_i = 0$ and $i = 1, \ldots, M$.

Three groups of randomly generated instances of difficult knapsack problems (KP) have been constructed to test the QEAs. In all instances the weights are uniformly distributed in a given interval. The profits are expressed as a function of the weights, yielding the specific properties of each group. Eight different problem instances for each group of KP have been constructed. Four different capacities of the knapsack have been considered, namely, 10%, 5%, 2%, and 1% of the total weight of all the items taken together. The number of items available for selection in this study is 200 and 5000 items.

4.3.1. Multiple Strongly Correlated Instances $mstr(k_1, k_2, d)$. They are constructed as a combination of two sets of strongly correlated instances, which have profits $\mathrm{pt}_j := \mathrm{wt}_j + k_m$ where $k_m, m = 1, 2$, is different for the two sets. The multiple strongly correlated instances $\mathrm{mstr}(k_1, k_2, d)$ have been generated in this work as follows: the weights of the M items are randomly distributed in $[1, 1000]$. If the weight wt_j is divisible by $d = 6$, then we set the profit $\mathrm{pt}_j := \mathrm{wt}_j + k_1$; otherwise, set it to $\mathrm{pt}_j := \mathrm{wt}_j + k_2$. The weights wt_j in the first group (i.e., where $\mathrm{pt}_j = \mathrm{wt}_j + k_1$) will all be multiples of d, so that using

Table 4: Comparative study between PQEA, CQEA, and FQEA using statistical results on 0-1 knapsack problem with multiple strongly correlated data instances.

% of total weight	Number of items (M)	Algo.	Best	Worst	Average	Median	Std	Average NFE
10%	200	PQEA	24222	23423	23965	24022	214.0	112880
		CQEA	24319	23722	23991	24021	136.2	127532
		FQEA	**24322**	**24220**	**24314**	**24321**	**25.5**	**73515**
	5000	PQEA	559926	551617	555315	555546	1985.7	499348
		CQEA	557944	550326	554411	554773	2015.7	498905
		FQEA	**592810**	**590092**	**591494**	**591494**	**806.6**	**498223**
5%	200	PQEA	15008	14605	14804	14857	127.4	139992
		CQEA	**15108**	14608	14871	14907	134.8	121588
		FQEA	**15108**	**15008**	**15104**	**15108**	**18.3**	**91160**
	5000	PQEA	334680	328120	331565	331569	1509.3	499008
		CQEA	330892	325231	327619	327652	1524.9	498883
		FQEA	**363587**	**359984**	**362343**	**362450**	**992.1**	**498093**
2%	200	PQEA	8398	7801	8147	8199	182.8	165077
		CQEA	**8400**	7601	8154	8149	165.2	172823
		FQEA	**8400**	**8300**	**8321**	**8301**	**40.2**	**137615**
	5000	PQEA	174665	168139	171836	171895	1462.0	499140
		CQEA	169071	162886	166073	166359	1654.3	498625
		FQEA	**197443**	**192821**	**195912**	**196074**	**1118.5**	**498843**
1%	200	PQEA	5497	4899	5285	5299	142.6	291117
		CQEA	5497	4899	5334	5396	139.4	297235
		FQEA	**5498**	**5399**	**5484**	**5497**	**34.0**	**255655**
	5000	PQEA	104861	101268	103519	103694	999.1	498967
		CQEA	100787	95661	98035	98054	1226.2	498928
		FQEA	**124745**	**122104**	**123315**	**123145**	**670.8**	**499153**

only these weights can at most use $d[C/d]$ of the capacity; therefore, in order to obtain a completely filled knapsack, some of the items from the second distribution will also be included. Computational experiments have shown that very difficult instances could be obtained with the parameters mstr (300, 200, and 6) [30].

The results of testing of all the three algorithms on 0-1 knapsack problem with multiple strongly correlated data instances have been presented in Table 4. A total of thirty independent runs of each algorithm were executed and the Best, Worst, Average, and Median and standard deviation (Std) of the fitness along with average number of function evaluations (NFE) were recorded. The maximum number of generations was ten thousand. FQEA has outperformed PQEA and CQEA on all the problem instances as indicated by the statistical results. CQEA has performed better than PQEA on problems with smaller number of items but PQEA has performed better than CQEA on problems with larger number of items.

Figures 19, 20, 21, 22, 23, 24, 25, and 26 show relative convergence rate of the QEAs for 0-1 knapsack problem with multiple strongly correlated data instances. The convergence graphs have been plotted between the number of generations and the objective function values of the three QEAs. The rate of convergence of PQEA is fastest during the early part of the search in most of the problem instances; however, FQEA was able to overtake both the QEAs in later part of the search process. CQEA has been the slowest of the three QEAs in most of the instances except when the capacity of the knapsack is small; that is, it has performed better than PQEA in such problem instances. PQEA has performed better than CQEA on all other problem instances. FQEA has been slow initially but has performed better than PQEA and CQEA on all the problem instances. The reason for poor performance of FQEA during the initial part of the search process is due to slow dispersion of information in FQEA as compared to the other QEAs, which enables FQEA to explore the solution space more comprehensively before reaching near a global optimum. In fact, slow dispersion of information helps it to reach near the global optimum as it avoids getting trapped in a local optimum. The dispersion of information is quickest in case of PQEA, which helps it to outperform CQEA and FQEA during the initial part of search process but also causes it to get trapped in a local optimum. The dispersion of information is slower in CQEA as compared to PQEA but CQEA has been able to reach near best solution found by PQEA in the later part of the search and in some cases CQEA has outperformed PQEA. Overall, FQEA has performed better than PQEA and CQEA on all the instances of 0-1 knapsack problem with multiple strongly correlated data distribution. Therefore,

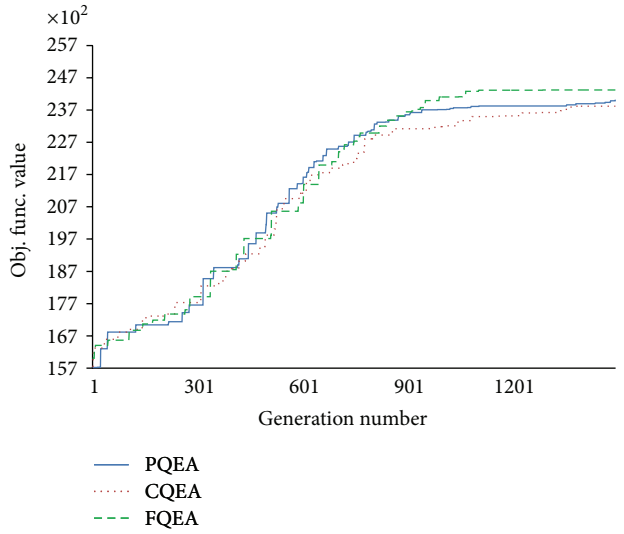


Figure 19: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with multiple strongly correlated data instances with number of items, M, being 200 and capacity, C, being 10% of total weight.

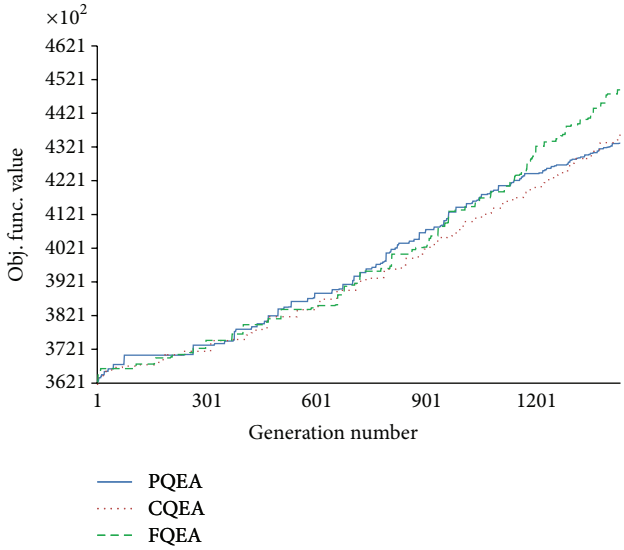


Figure 20: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with multiple strongly correlated data instances with number of items, M, being 5000 and capacity, C, being 10% of total weight.

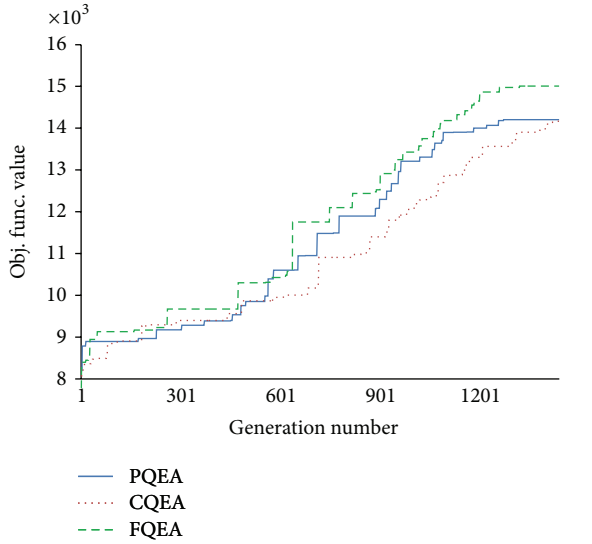


Figure 21: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with multiple strongly correlated data instances with number of items, M, being 200 and capacity, C, being 5% of total weight.

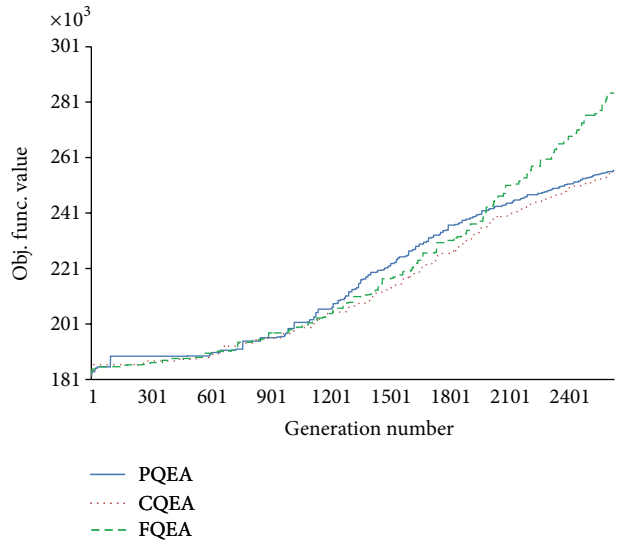


Figure 22: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with multiple strongly correlated data instances with number of items, M, being 5000 and capacity, C, being 5% of total weight.

the slow rate of dispersion of information in the fine-grained model had helped FQEA to perform better than QEAs with the other population models in 0-1 knapsack problem with multiple strongly correlated data instances.

4.3.2. Profit Ceiling Instances pceil(a). These instances have profits of all items as multiples of a given parameter a. The weights of the M items are randomly distributed in $[1, 1000]$, and their profits are set to $\mathrm{pt}_j = a[\mathrm{wt}_j/a]$. The parameter, a, has been experimentally chosen as $a = 3$, as this resulted in sufficiently difficult instances [30].

The results of testing of all the three algorithms on 0-1 knapsack problem with profit ceiling distribution instances have been presented in Table 5. A total of thirty independent runs of each algorithm were executed and the Best, Worst, Average, and Median and standard deviation (Std) of the fitness along with the average number of function evaluations (NFE) were recorded. The maximum number of generations was ten thousand. FQEA has outperformed both

Table 5: Comparative study between PQEA, CQEA, and FQEA using statistical results on 0-1 knapsack problem with profit ceiling data instances.

% of total weight	Number of items (M)	Algo.	Best	Worst	Average	Median	Std	Average NFE
10.00%	200	PQEA	10113	10095	10104	10104	5.3	208817
		CQEA	10116	10092	10105	10104	5.1	256945
		FQEA	**10137**	**10122**	**10130**	**10131**	**3.8**	**165498**
	5000	PQEA	249807	249684	249753	249750	33.2	491542
		CQEA	249723	249627	249688	249687	**22.2**	491412
		FQEA	**250521**	**250395**	**250474**	**250478**	31.7	**485267**
5.00%	200	PQEA	5070	5049	5060	5061	5.6	190365
		CQEA	5070	5049	5061	5061	5.5	198847
		FQEA	**5091**	**5073**	**5082**	**5082**	**4.5**	**127020**
	5000	PQEA	125028	124944	124993	124994	23.9	483715
		CQEA	125004	124929	124964	124962	**20.3**	**482153**
		FQEA	**125553**	**125448**	**125504**	**125505**	26.4	488277
2.00%	200	PQEA	2040	2025	2030	2028	**4.1**	238790
		CQEA	2040	2022	2032	2033	4.5	239368
		FQEA	**2052**	**2034**	**2044**	**2043**	4.7	**153933**
	5000	PQEA	50103	50040	50070	50066	19.4	475055
		CQEA	50097	50028	50059	50066	**17.1**	**477915**
		FQEA	**50436**	**50346**	**50389**	**50390**	20.0	485703
1.00%	200	PQEA	1023	**1014**	1017	1017	2.9	**197130**
		CQEA	1026	**1014**	1018	1017	**2.7**	252298
		FQEA	**1032**	**1014**	**1023**	**1023**	4.5	243558
	5000	PQEA	25086	25029	25060	25059	13.7	472835
		CQEA	25077	25026	25054	25053	**11.8**	**471443**
		FQEA	**25329**	**25236**	**25281**	**25278**	21.8	491103

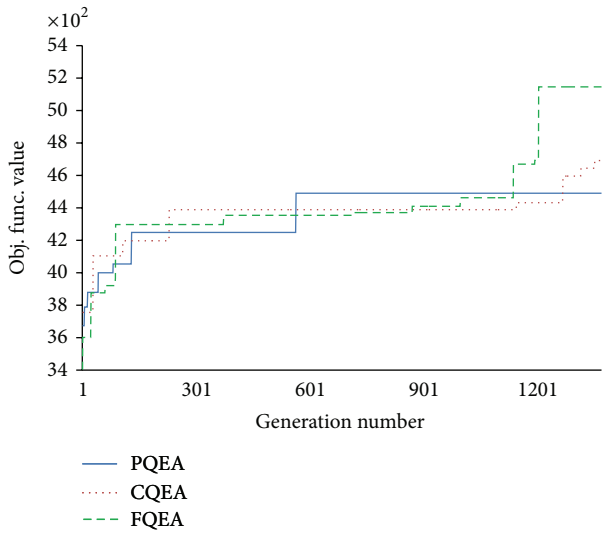


Figure 23: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with multiple strongly correlated data instances with number of items, M, being 200 and capacity, C, being 2% of total weight.

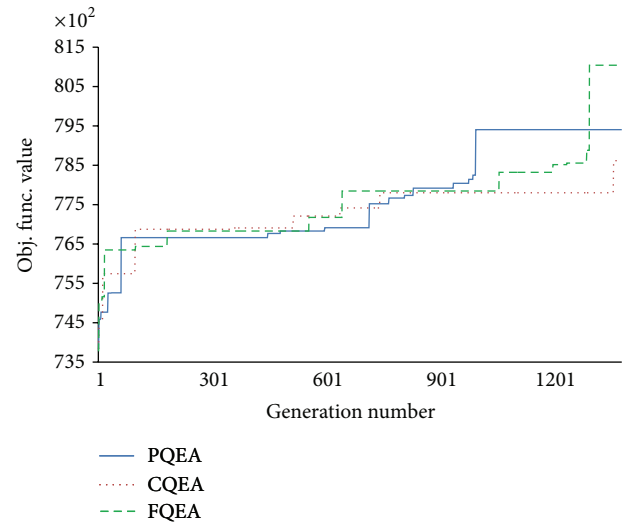


Figure 24: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with multiple strongly correlated data instances with number of items, M, being 5000 and capacity, C, being 2% of total weight.

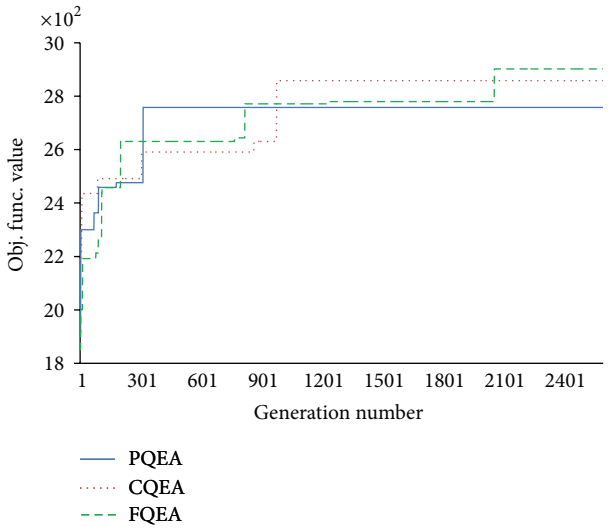


Figure 25: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with multiple strongly correlated data instances with number of items, M, being 200 and capacity, C, being 1% of total weight.

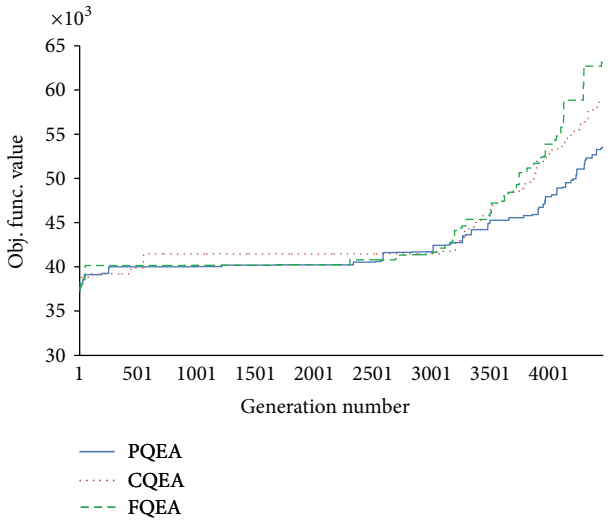


Figure 26: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with multiple strongly correlated data instances with number of items, M, being 5000 and capacity, C, being 1% of total weight.

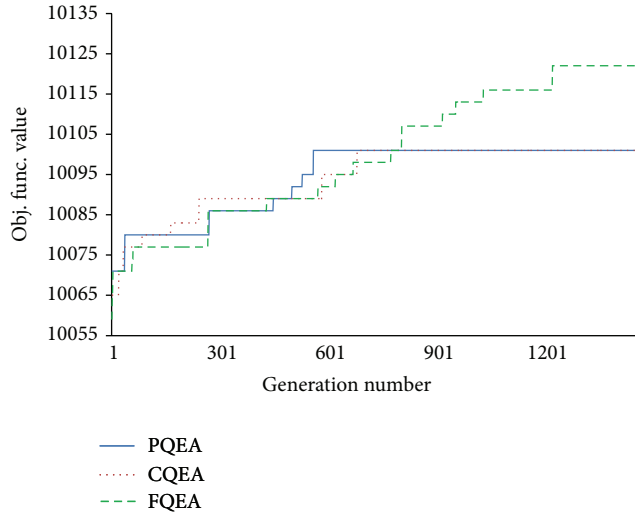


Figure 27: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with profit ceiling data instances with number of items, M, being 200 and capacity, C, being 10% of total weight.

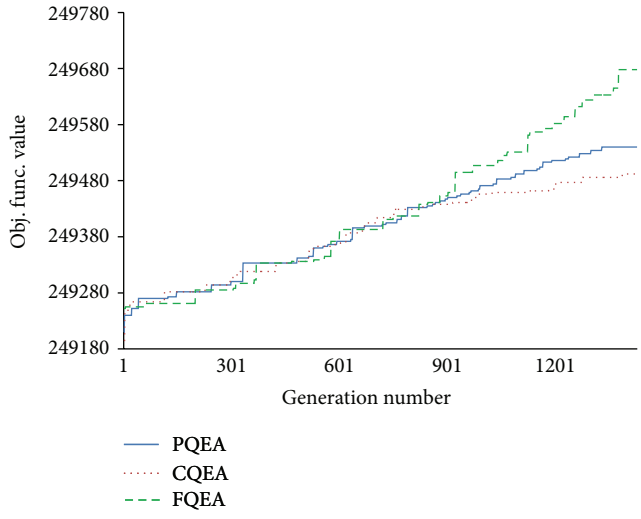


Figure 28: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with profit ceiling data instances with number of items, M, being 5000 and capacity, C, being 10% of total weight.

the other QEAs on all the problem instances as indicated by the statistical results. CQEA has performed better than PQEA on problems with smaller number of items but PQEA has performed better than CQEA on problems with larger number of items.

Figures 27, 28, 29, 30, 31, 32, 33, and 34 show relative convergence rate of the QEAs for 0-1 knapsack problem with profit ceiling data instances. The convergence graphs have been plotted between the number of generations and the objective function values of the three QEAs. The rate of convergence of PQEA is fastest during the early part of the search process in most of the problem instances; however, FQEA was able to overtake both the QEAs during later part of the search process. CQEA has been the slowest of the three QEAs in most of the instances except when capacity of knapsack is small; that is, CQEA has performed better than PQEA in such problem instances. PQEA has performed better than CQEA on the other problem instances. FQEA has been slow initially but has performed better than both PQEA and CQEA on all the problem instances. The reason for poor performance of FQEA initially is due to the slow dispersion of information as compared to the other QEAs, which enables FQEA to explore the solution space more comprehensively before reaching near a global optimum. In fact, slow dispersion of information helps FQEA to reach

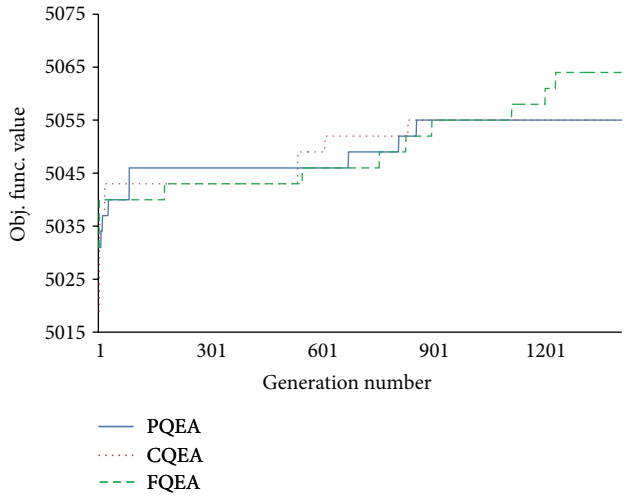


Figure 29: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with profit ceiling data instances with number of items, M, being 200 and capacity, C, being 5% of total weight.

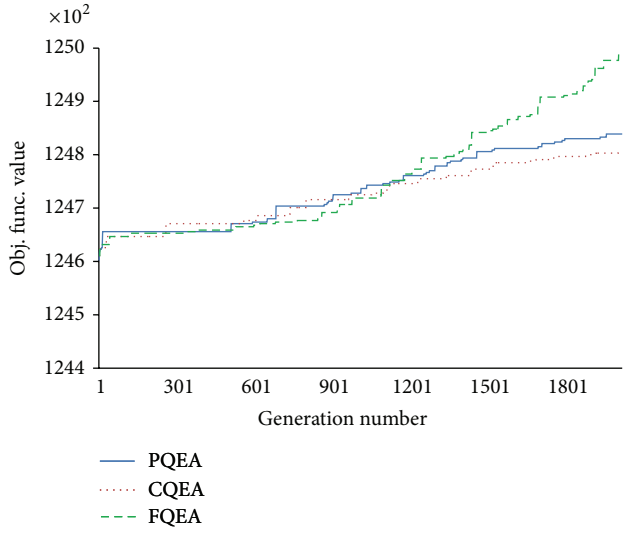


Figure 30: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem profit ceiling data instances with number of items, M, being 5000 and capacity, C, being 5% of total weight.

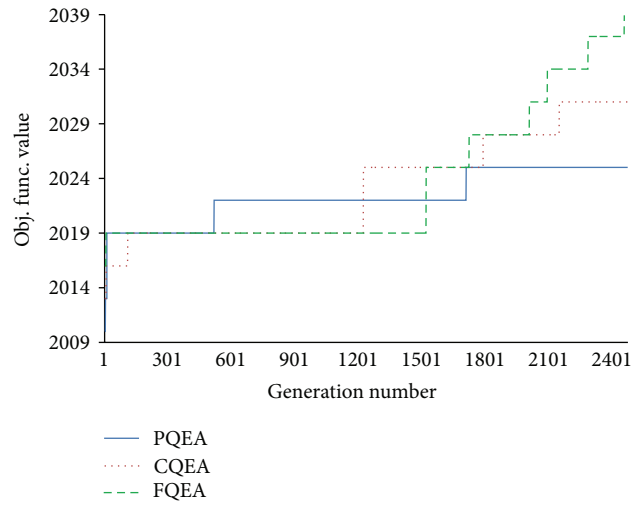


Figure 31: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with profit ceiling data instances with number of items, M, being 200 and capacity, C, being 2% of total weight.

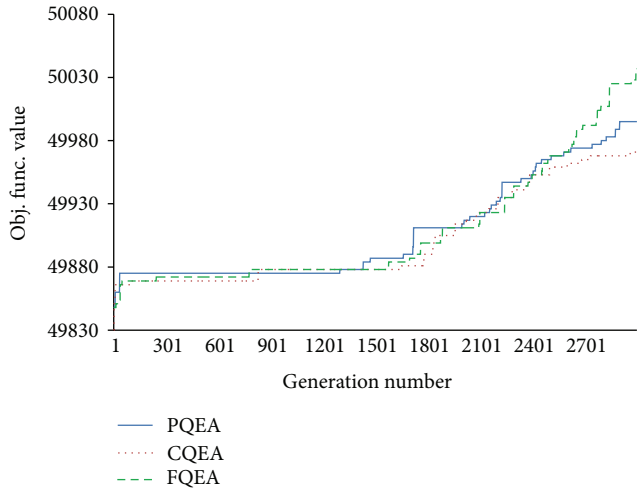


Figure 32: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with profit ceiling data instances with number of items, M, being 5000 and capacity, C, being 2% of total weight.

near a global optimum as it avoids getting trapped in a local optimum. The dispersion of information is quickest in the case of PQEA, which helps it to outperform CQEA and FQEA in the initial part of the search process, but also causes it to get trapped in a local optimum. The dispersion of information is slower in CQEA as compared to PQEA but CQEA has been able to reach near the best solution found by PQEA in the later part of the search process in some cases. Overall, FQEA has performed better than PQEA and CQEA on all the instances of 0-1 knapsack problem with profit ceiling data distribution. Therefore, the slow rate of dispersion of information in the fine-grained model had helped FQEA to perform better than the QEAs with other population models in 0-1 knapsack problems with profit ceiling data distribution.

4.3.3. Circle Instances circle(d_c). These instances have the profit of their items as function of the weights forming an arc of a circle (actually an ellipsis). The weights are uniformly distributed in $[1, 1000]$ and for each weight wt_i the corresponding profit is chosen to be $\text{pt}_i = d_c\sqrt{2000^2 - (\text{wt}_i - 2000)^2}$. Experimental results have shown in [30] that difficult instances appeared by choosing $d_c = 2/3$ which was chosen for testing in this work.

The results of testing of all the three algorithms on 0-1 knapsack problem with circle data instances have been presented in Table 6. A total of thirty independent runs of each algorithm were executed and the Best, Worst, Average, and Median and standard deviation (Std) of the fitness along with average number of function evaluations (NFE) were

Table 6: Comparative study between PQEA, CQEA, and FQEA using statistical results on 0-1 knapsack problem with circle data instances.

% of total weight	Number of items (M)	Algo.	Best	Worst	Average	Median	Std	Average NFE
10.00%	200	PQEA	31583	30647	31180	31221	193.8	119295
		CQEA	**31587**	30811	31265	31257	206.0	116388
		FQEA	**31587**	**31581**	**31586**	**31587**	**1.8**	**73537**
	5000	PQEA	708392	697953	703964	704143	2420.4	499152
		CQEA	703183	692142	697451	696994	2945.7	498935
		FQEA	**764159**	**758628**	**761728**	**761963**	**1526.5**	**498415**
5.00%	200	PQEA	**19049**	18142	18682	18747	196.2	134142
		CQEA	19046	18462	18772	18774	125.7	138005
		FQEA	**19049**	**19046**	**19048**	**19049**	**0.7**	**84348**
	5000	PQEA	411918	405428	408012	407759	1753.3	499310
		CQEA	401861	393326	398180	398439	1937.1	**498628**
		FQEA	**452412**	**446621**	**449780**	**449917**	**1296.1**	498718
2.00%	200	PQEA	9613	9044	9413	9420	159.1	214528
		CQEA	**9621**	9118	9477	9519	138.0	198957
		FQEA	**9621**	**9558**	**9611**	**9621**	**21.1**	**139205**
	5000	PQEA	198546	194542	196730	196844	**949.3**	498778
		CQEA	191986	186153	188308	188099	1530.3	**498763**
		FQEA	**223217**	**219368**	**220820**	**220720**	1080.1	498860
1.00%	200	PQEA	**5802**	5198	5534	5555	152.4	303258
		CQEA	**5802**	5210	5591	5613	142.9	318865
		FQEA	**5802**	**5638**	**5784**	**5802**	**46.7**	**258733**
	5000	PQEA	112605	107915	110401	110440	1036.4	499048
		CQEA	107044	102173	104741	104807	1248.8	**498433**
		FQEA	**129271**	**126639**	**128167**	**128285**	**736.5**	499162

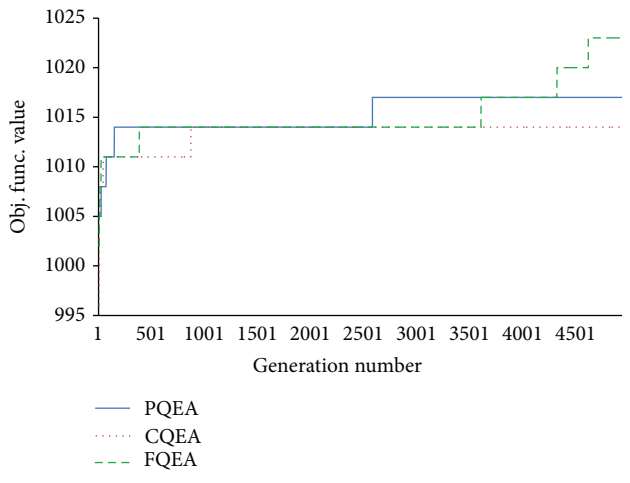


Figure 33: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem profit ceiling data instances with number of items, M, being 200 and capacity, C, being 1% of total weight.

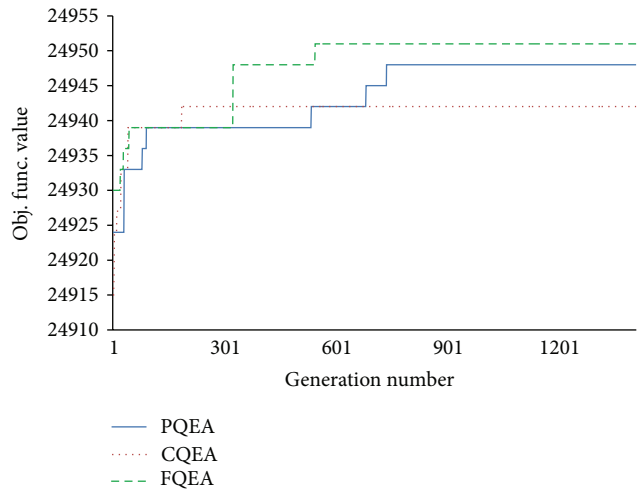


Figure 34: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with profit ceiling data instances with number of items, M, being 5000 and capacity, C, being 1% of total weight.

recorded. The maximum number of generations was ten thousand. FQEA has outperformed both PQEA and CQEA on all the problem instances as indicated by the statistical results. CQEA has performed better than PQEA on problems with smaller number of items but PQEA has performed better than CQEA on problems with larger number of items.

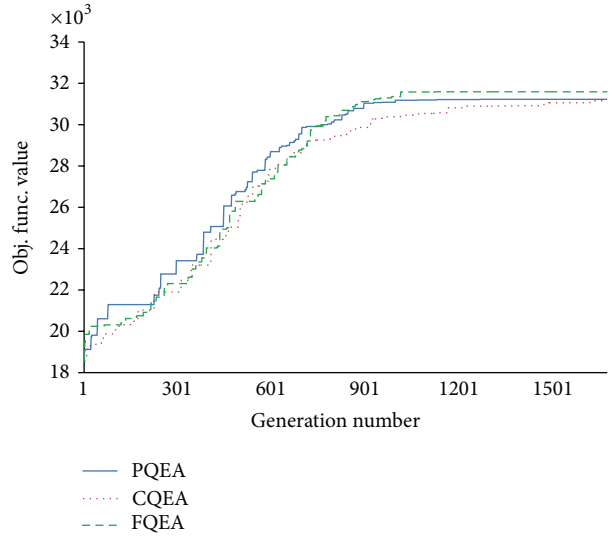


Figure 35: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with circle data instances with number of items, M, being 200 and capacity, C, being 10% of total weight.

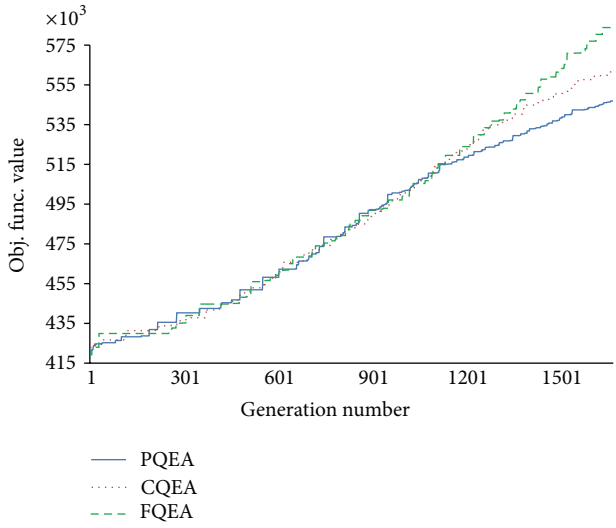


Figure 36: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with circle data instances with number of items, M, being 5000 and capacity, C, being 10% of total weight.

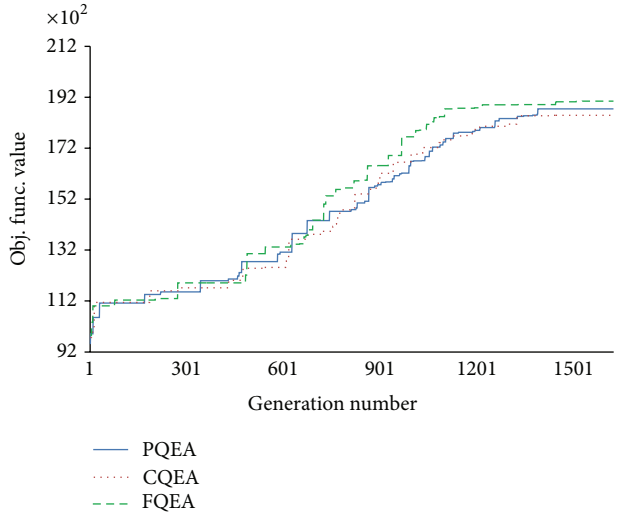


Figure 37: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with circle data instances with number of items, M, being 200 and capacity, C, being 5% of total weight.

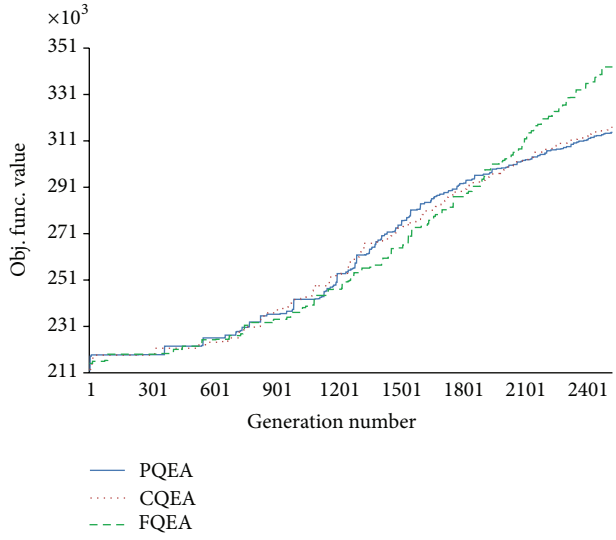


Figure 38: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem circle data instances with number of items, M, being 5000 and capacity, C, being 5% of total weight.

Figures 35, 36, 37, 38, 39, 40, 41, and 42 show relative convergence rate of the QEAs for 0-1 knapsack problem with circle data distribution instances. The convergence graphs have been plotted between the number of generations and the objective function values of the three QEAs. The rate of convergence of PQEA is fastest during the early part of the search process in some problem instances; however, FQEA was able to overtake both the QEAs during later part of the search process. CQEA has been the slowest of the three QEAs in most of the problem instances. CQEA has performed better than PQEA on most of the problem instances. FQEA has been slow initially but has performed better than both PQEA and CQEA on all the problem instances. The reason for poor performance of FQEA initially is due to the slow dispersion of information as compared to the other QEAs, which enables it to explore the solution space more comprehensively before reaching near a global optimum. In fact, slow dispersion of information helps it to reach near the global optimum as it avoids getting trapped in a local optimum. The dispersion of information is quickest in case of PQEA, which helps it to outperform CQEA and FQEA in initial part of the search process, but also causes it to get trapped in a local optimum. The dispersion of information is slower in CQEA as compared to PQEA but it has been able to reach near the best solution found by PQEA in the later part of the search in some cases. CQEA has also outperformed

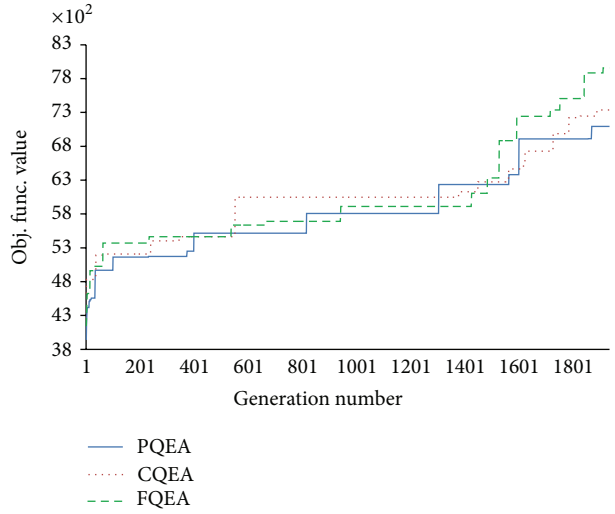


Figure 39: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with circle data instances with number of items, M, being 200 and capacity, C, being 2% of total weight.

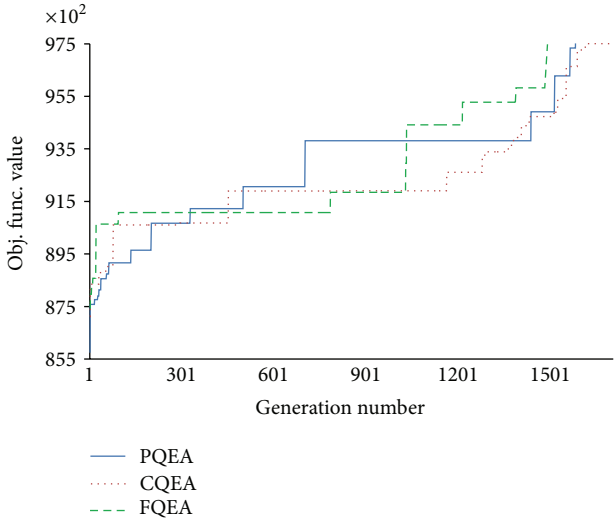


Figure 40: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with circle data instances with number of items, M, being 5000 and capacity, C, being 2% of total weight.

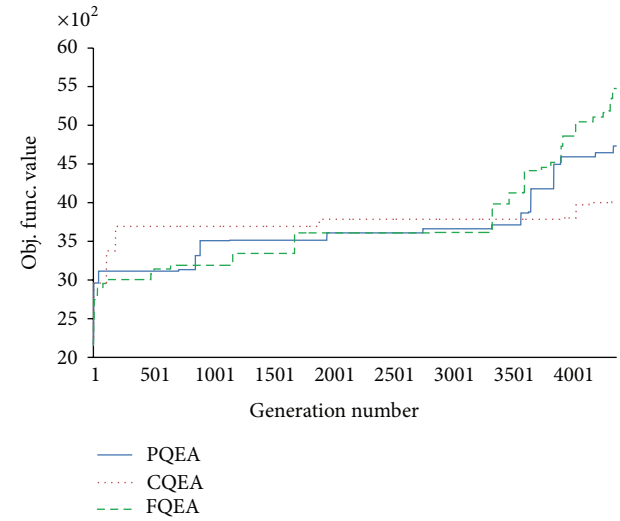


Figure 41: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem circle data instances with number of items, M, being 200 and capacity, C, being 1% of total weight.

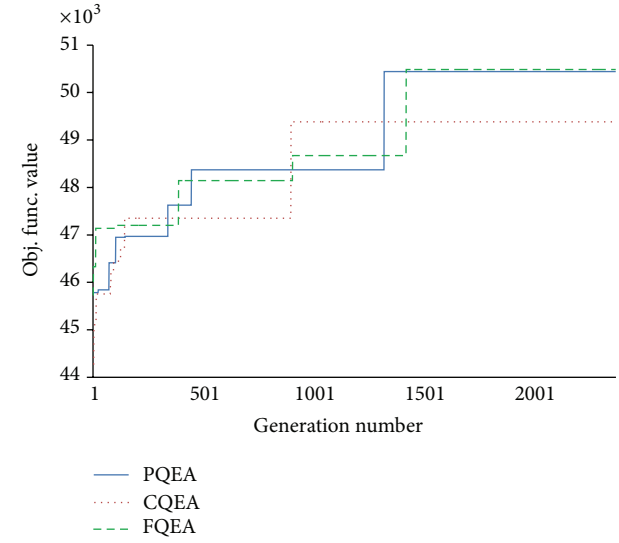


Figure 42: Convergence graph of PQEA, CQEA, and FQEA on 0-1 knapsack problem with circle data instances with number of items, M, being 5000 and capacity, C, being 1% of total weight.

PQEA in some cases. Overall, FQEA has performed better than PQEA and CQEA on all the instances of 0-1 knapsack problem with circle data distribution. Therefore, the slow rate of dispersion of information in the fine-grained model had helped FQEA to perform better than the QEAs with other population models in 0-1 knapsack problem with circle data distribution.

4.4. Comparative Study. A comparative study has been performed between FQEA and recently proposed "state-of-the-art" algorithm known as "hybrid cuckoo search algorithm with improved shuffled frog leaping algorithm" (CSISFLA) [31]. It was shown in [31] that CSISFLA has performed better than genetic algorithm [3], Differential Evolution Algorithm [32, 33], and Cuckoo Search [34] on some 0-1 knapsack problem instances. The size of the knapsack is very large as it is 75% of the total weight of all the items taken together. Table 7 empirically compares the performance of FQEA and CSISFLA on KP-1, KP-2, KP-3, and KP-7, which are 0-1 knapsack problem instances with uncorrelated weight and profit instances given in [31]. The duration of execution of FQEA (compiled with Visual Studio 6) was five seconds for KP-1, KP-2, and KP-3 and eight seconds on a computer with AMD Athlon 7750 Dual-Core, 2.71 GHz, 1.75 GB RAM running under Windows XP, which was a similar machine to that used for running CSISFLA in [31]. The target capacity,

Table 7: Comparative study between CSISFLA and FQEA on 0-1 knapsack problem with uncorrelated distribution.

Instance	Algorithm	Best	Worst	Mean	Median	Std
KP-1	CSISFLA	7475	7467	7473	7474	**1.6**
	FQEA	**7498**	**7473**	**7494**	**7497**	7.7
KP-2	CSISFLA	9865	9837	9856	9858	7.2
	FQEA	**10040**	**10039**	**10040**	**10040**	**0.5**
KP-3	CSISFLA	15327	15248	15297	15302	18.5
	FQEA	**15378**	**15370**	**15375**	**15376**	**2.5**
KP-7	CSISFLA	**60779**	**60264**	**60443**	**60420**	130.6
	FQEA	59596	59338	59427	59409	**76.9**

Table 8: Comparative study between CSISFLA and FQEA on 0-1 knapsack problems with the number of items being 1200.

Instance	Algorithm	Best	Worst	Mean	Median	Std
KP-7	CSISFLA	60779	60264	60443	60420	130.6
	FQEA	**61831**	**61811**	**61820**	**61820**	**6.1**
KP-14	CSISFLA	52403	52077	52267	52264	86.2
	FQEA	**52538**	**52518**	**52526**	**52526**	**5.8**
KP-19	CSISFLA	60562	60539	60549	60550	**5.7**
	FQEA	**60570**	**60550**	**60560**	**60560**	6.0
KP-24	CSISFLA	72151	72070	72112	72111	21.2
	FQEA	**72232**	**72192**	**72202**	**72192**	**13.7**
KP-29	CSISFLA	51399	51390	51396	51396	**3.1**
	FQEA	**51405**	**51390**	**51398**	**51399**	4.2
KP-34	CSISFLA	84244	84099	84175	84181	38.4
	FQEA	**85090**	**85016**	**85041**	**85035**	**25.7**

total capacity, and total value of the items in each problem are the same as those given in [31].

Results in Table 7 show that FQEA outperforms CSISFLA on KP-1, KP-2, and KP-3, but CSISFLA performs better than FQEA on KP-7. This shows that FQEA is performing better on small dimension problems compared to CSISFLA but CSISFLA is performing better than FQEA on large dimension problems. In order to find the reason for poor performance of FQEA on large dimension problems, an investigation was made on the conditions of comparative study. It was found that KP-1 has 150 items, KP-2 has 200 items, KP-3 has 300 items, and KP-7 has 1200 items. Thus, by keeping total running time of five seconds for KP-1 to KP-3 shows that it is either more for KP-1 or less for KP-3, because the problem size in KP-3 is double that of KP-1. So let us assume that if 5 seconds of run time was adequate for KP-3, then, it is considerably more for KP-1. However, even when CSISFLA has evolved for more time in case of KP-1, it has produced inferior result as compared to FQEA, showing that CSISFLA has a tendency of getting trapped in suboptimal region. The search process of FQEA is slow initially as it explores the search space in a more comprehensive way due to slow dissemination information in its population structure as compared to other algorithms. In case of KP-7, the problem size is eight times that of KP-1. Thus, if FQEA is evolved for forty seconds instead of eight seconds, it should produce better results than CSISFLA. It can be argued that CSISFLA may produce better result if it evolved for forty seconds, but, as we have seen in KP-1, evolving for more duration is not necessarily helping CSISFLA. Table 8 shows the results of FQEA with forty seconds of execution time on KP-7, KP-14, KP-19, KP-24, KP-29, and KP-34 along with the results of CSISFLA given in [31]. Table 8 shows that FQEA is able to produce better results when evolved for longer duration. However, CSISFLA produces better result in large dimension problem instances than FQEA, when evolved for eight seconds only.

5. Conclusions

Quantum-inspired evolutionary algorithm are a type of evolutionary algorithms, which have been designed by integrating probabilistic bit representation, superposition, and measurement principles in evolutionary framework for computationally solving difficult optimization problems. The QEA in [7] has been developed on coarse-grained population model, which has been subsequently modified into panmictic model, vQEA, and is shown to be better than QEA in solving some problems [8]. This motivated further investigations of the effect of fine-grained population structure on the performance of QEA [35]. The experimental testing has shown that fine-grained model improves the performance of QEA in comparison to the other models on COUNTSAT problem instances, *P*-PEAKS problem instances, and three difficult knapsack problem instances. Thus, the contribution of this work is twofold; namely, the first is the critical examination of the population structures employed in QEA and the implementation of QEA on the two-dimensional toroidal grid for fair comparison of the effect of all the three population models on QEA. A comparative study was also performed with the "state-of-the-art" hybrid cuckoo search algorithm [31], which showed that FQEA is slow but produces better solutions. Future endeavors will be made to further improve the speed of convergence in FQEA for solving benchmark as well as real world search and optimization problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are thankful to anonymous reviewers for giving insightful comments, which helped in improving the paper. Nija Mani is thankful to her department and UGC for supporting her Ph.D. through Meritorious Student's Fellowship.

References

[1] X.-S. Yang, S. Koziel, and L. Leifsson, "Computational optimization, modelling and simulation: recent trends and challenges," in *Proceedings of the 13th Annual International Conference on*

Computational Science (ICCS '13), vol. 18, pp. 855–860, June 2013.

[2] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 443–462, 2002.

[3] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, London, UK, 3rd edition, 1996.

[4] Z. Michalewicz and D. B. Fogel, *How To Solve It: Modern Heuristics*, Springer, Berlin, Germany, 2nd edition, 2004.

[5] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York, NY, USA, 1989.

[6] A. Mani and C. Patvardhan, "Analysis of photoluminescence measurement data from interdiffused quantum wells by real coded quantum inspired evolutionary algorithm," in *Proceedings of the 13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research*, Proceedings of Science, Jaipur, India, February 2010.

[7] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, 2002.

[8] M. D. Platel, S. Sehliebs, and N. Kasabov, "A versatile quantum-inspired evolutionary algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 423–430, Singapore, September 2007.

[9] L. Kliemann, O. Kliemann, C. Patvardhan, V. Sauerland, and A. Srivastav, "A new QEA computing near-optimal low-discrepancy colorings in the hypergraph of arithmetic progressions," in *Proceedings of 12th International Symposium Experimental Algorithms*, vol. 7933 of *Lecture Notes in Computer Science*, pp. 67–78, June 2013.

[10] D. H. Kim, Y. H. Yoo, S. J. Ryu, W. Y. Go, and J. H. Kim, "Automatic color detection for MiroSOT using quantum-inspired evolutionary algorithm," in *Intelligent Robotics Systems: Inspiring the NEXT Communications in Computer and Information Science*, vol. 376, pp. 11–20, 2013.

[11] A. C. Kumari, K. Srinivas, and M. P. Gupta, "Software requirements selection using quantum-inspired elitist multi-objective evolutionary algorithm," in *Proceedings of the 1st International Conference on Advances in Engineering, Science and Management (ICAESM '12)*, pp. 782–787, March 2012.

[12] H. Lei and K. Qin, "Quantum-inspired evolutionary algorithm for analog test point selection," *Analog Integrated Circuits and Signal Processing*, vol. 75, no. 3, pp. 491–498, 2013.

[13] Y. Li, S. Feng, X. Zhang, and L. Jiao, "SAR image segmentation based on quantum-inspired multiobjective evolutionary clustering algorithm," *Information Processing Letters*, vol. 114, no. 6, pp. 287–293, 2014.

[14] A. Mani and C. Patvardhan, "A hybrid quantum evolutionary algorithm for solving engineering optimization problems," *International Journal of Hybrid Intelligent System*, vol. 7, no. 3, pp. 225–235, 2010.

[15] A. Mani and C. Patvardhan, "An improved model of ceramic grinding process and its optimization by adaptive Quantum inspired evolutionary algorithm," *International Journal of Simulations: Systems Science and Technology*, vol. 11, no. 3, pp. 76–85, 2012.

[16] C. Patvardhan, A. Narain, and A. Srivastav, "Enhanced quantum evolutionary algorithms for difficult knapsack problems," in *Proceedings of International Conference on Pattern Recognition and Machine Intelligence*, vol. 4815 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2007.

[17] G. Zhang, "Quantum-inspired evolutionary algorithms: a survey and empirical study," *Journal of Heuristics*, vol. 17, pp. 303–351, 2011.

[18] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 126–142, 2005.

[19] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithms with a new termination criterion, Hε gate, and two-phase scheme," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 156–169, 2004.

[20] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceeding of the Congress on Evolutionary Computation (CEC '02)*, vol. 2, pp. 1671–1676, May 2002.

[21] E. Alba and J. M. Troya, "Improving flexibility and efficiency by adding parallelism to genetic algorithms," *Statistics and Computing*, vol. 12, no. 2, pp. 91–114, 2002.

[22] P. Spiessens and B. Manderick, "A massively parallel genetic algorithm," in *Proceeding of 4th International Conference on Genetic Algorithms*, R. Bclew and L. Booker, Eds., pp. 279–286, 1991.

[23] K. W. C. Ku, M. W. Mak, and W. C. Siu, "Adding learning to cellular genetic algorithms for training recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 2, pp. 239–252, 1999.

[24] G. Folino, C. Pizzuti, and G. Spezzano, "Parallel hybrid method for SAT that couples genetic algorithms and local search," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 323–334, 2001.

[25] D. A. Sofge, "Prospective algorithms for quantum evolutionary computation," in *Proceedings of the 2nd Quantum Interaction Symposium (QI '08)*, College Publications, Oxford, UK, 2008, http://arxiv.org/ftp/arxiv/papers/0804/0804.1133.pdf.

[26] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2006.

[27] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 61–66, May 1996.

[28] S. Droste, T. Jansen, and I. Wegener, "A natural and simple function which is hard for all evolutionary algorithms," in *Proceedings of the 26th Annual Conference of the IEEE Electronics Society (IECON '00)*, pp. 2704–2709, Nagoya, Japan, October 2000.

[29] K. de Jong, M. Potter, and W. Spears, "Using problem generators to explore the effects of epistasis," in *Proceedings of the 7th International Conference on Genetic Algorithms*, T. Bäck, Ed., pp. 338–345, 1997.

[30] D. Pisinger, "Where are the hard knapsack problems?" *Computers & Operations Research*, vol. 32, no. 9, pp. 2271–2284, 2005.

[31] Y. Feng, G.-G. Wang, Q. Feng, and X.-J. Zhao, "An effective hybrid cuckoo search algorithm with improved shuffled frog leaping algorithm for 0-1 Knapsack problems," *Computational Intelligence and Neuroscience*, vol. 2014, Article ID 857254, 17 pages, 2014.

[32] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[33] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1679–1696, 2011.

[34] A. Gherboudj, A. Layeb, and S. Chikhi, "Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm," *International Journal of Bio-Inspired Computation*, vol. 4, no. 4, pp. 229–236, 2012.

[35] N. Mani, G. Srivastava, A. K. Sinha, and A. Mani, "An evaluation of cellular population model for improving quantum-inspired evolutionary algorithm," in *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation (GECCO '12)*, pp. 1437–1438, Philadelphia, Pa, USA, July 2012.