How to Exploit Fitness Landscape Properties of Timetabling Problem: A New operator for Quantum Evolutionary Algorithm

Mohammad Hassan Tayarani Najaran[a]

[a] University of Hertfordshire, Hatfield, UK.

ARTICLE INFO

Keywords:
Timetabling-Colouring
fitness landscape analysis
Quantum Evolutionary Algorithms.

Abstract

The fitness landscape of the timetabling problems is analyzed in this paper to provide some insight into the properties of the problem. The analyses suggest that the good solutions are clustered in the search space and there is a correlation between the fitness of a local optimum and its distance to the best solution. Inspired by these findings, a new operator for Quantum Evolutionary Algorithms is proposed which, during the search process, collects information about the fitness landscape and tried to capture the backbone structure of the landscape. The knowledge it has collected is used to guide the search process towards a better region in the search space. The proposed algorithm consists of two phases. The first phase uses a tabu mechanism to collect information about the fitness landscape. In the second phase, the collected data are processed to guide the algorithm towards better regions in the search space. The algorithm clusters the good solutions it has found in its previous search process. Then when the population is converged and trapped in a local optimum, it is divided into sub-populations and each sub-population is designated to a cluster. The information in the database is then used to reinitialize the q-individuals, so they represent better regions in the search space. This way the population maintains diversity and by capturing the fitness landscape structure, the algorithm is guided towards better regions in the search space. The algorithm is compared with some state-of-the-art algorithms from PATAT competition conferences and experimental results are presented.

1. Introduction

To design successful optimization algorithms requires an understanding of the structure of the problems. Thus, there has been a great interest in the analysis of the fitness landscape of many optimization problems. The concept of the fitness landscape, introduced by Sewell Wright Wright (1932) to demonstrate the dynamics of biological evolutionary optimization, has been useful for the analysis and understanding of the evolutionary algorithm's behavior. The concept has been studied in a variety of fields including in physics Mézard et al. (1987); Hartmann and Weight (2005) and in the optimization community Huberman and T (1987); Cheeseman et al. (1991); Grover (1992); Hertz et al. (1994b). Research on landscape analysis in evolutionary algorithms started with the works presented in the early 1990s Manderick et al. (1991); Mathias and Whitley (1992); Forrest and Mitchell (1993); Jones and Forrest (1995). The first measure proposed for the roughness of the fitness landscape was the auto-correlation Weinberger (1990); Angel and Zissimopoulos (1998); Czogalla (2008). Then the fitness-distance correlation was proposed to measure problem difficulty Jones and Forrest (1995); Merz and B.Freisleben (2000); Lefticaru and Ipate (2008); Manderick et al. (1991).

The attempts continued with studying algebraic properties of the solutions in the landscape Grover (1992); Stadler (1995); Whitley et al. (2008); Chicano et al. (2010), modality (number of local optima) Horn and Goldberg (1995) and the fractal dimension of the fitness landscape Hoshino et al. (1998). Another way of analyzing the problem hardness is by studying the area in the landscape called "Olympus", in which the better local optima are located Verel et al. (2007); Vérel et al. (2008). Visualizing the fitness landscape, which is called fitness clouds is another method proposed to represent some properties of the fitness landscape Collard et al. (2007); Lu et al. (2011); Vanneschi et al. (2007), reflecting the problem hardness. The use of complex network analysis techniques has been proposed for studying fitness landscapes and problem difficulty in combinatorial optimization Daolio et al. (2012) and has recently been used to study the landscape of Quadratic Assignment Problem Daolio et al. (2010).

Fitness landscape analysis techniques are used to better understand the influence of genetic representations and associated variation operators in solving combinatorial optimization problems McCarthy (2008); Tavares et al. (2006, 2008); Riley and Ciesielski (2010). This understanding can provide useful information about the structure of the problem and the type of operators that are better for particular problems Newth and Brede (2006); Slany and Sekanina (2007); Merz and Freisleben (1998); Czogalla and Fink (2009). Furthermore, the study of fitness landscape can be useful in designing evolutionary algorithms or hybrid algorithms Moscato (1989); Moscato and Norman (1992); Qasem and Prügel-Bennett, since the landscape analysis can help us predict the performance of the proposed algorithms Shaowei and Qiuping (2007); Huanga et al. (2009). Some researchers use the landscape analysis to study some parameters of the evolutionary algorithms, like the population size Alander (1999), or some operators like mutation and crossover Mathias and Whitley (1992); Suzuki and Iwasa (1997), the recombination operators Hornby (1996), or the perturbation operator Martin et al. (1999). Some researchers have used the landscape analysis to explain why some algorithms, like local search algorithms Fonlupt et al. (1997), Memetic Algorithms Merz (2004) or metaheuristic algorithms based on local search Watson (2010), work better on particular landscapes.

m.tayaraninajaran@herts.ac.uk (M.H.T. Najaran)
ORCID(s):

Several works attempt to exploit the concept of the landscape and landscape analysis in developing new sets of algorithms for different problems. Fitness landscape analysis is used to propose Memetic Algorithms for Graph Bi-Partitioning problem Merz and Freisleben (1998), Resource Allocation problem Huang et al. (2009) and Maximum Satisfiability problem Zhang et al. (2003); Zhang (2004), or improving the performance of evolutionary algorithms by a landscape approximation Ratle (1998); Pošík and Franc (2007); Shen and He (2010). In a more recent work, the landscape analysis is used to propose a new population-based algorithm Qasem and Prügel-Bennett (2010).

During the last two decades many researchers have studied the landscape of optimisation problems including Travelling Salesman Mathias and Whitley (1992); Stadler and Schnabl (1992); Boese (1995); Whitley and Ochoa (2011); Whitley and Chicano (2012); Tayarani-N. and Prugel-Bennett (2016), Quadratic Assignment Merz and B.Freisleben (2000); Angel and Zissimopoulos (2001); Chicano et al. (2010); Tayarani-N. and Prügel-Bennett (2015), Knapsack Yoshizawa and Hashimoto (2000); Tavares et al. (2008), Max-Sat Weixiong and Zhang (2004); Sutton et al. (2009, 2010); Qasem and Prügel-Bennett (2010); Prugel-Bennett and Tayarani-Najaran (2011), graph drawing Lehn and Kuntz (2001), Graph-Colouring Hertz et al. (1994a); Hamiez and Hao (2001); Culberson and Gent (2001); Bouziri et al. (2009); Hertz et al. (1994b); Bouziri et al. (2011); Tayarani-N and Prügel-Bennett (2015); Tayarani-N. and Prugel-Bennett (2014), evolutionary antenna design Alander et al. (2002), flow-shop scheduling Czogalla and Fink (2011); Zhao et al. (2019) and Bayesian network structure Wu et al. (2011) problems.

There are many works that have attempted to solve the timetabling problems. This section provides an overview of recent works on timetabling problems. For earlier research in this field, the readers are referred to the survey on the approaches for university timetabling problem presented in Babaei et al. (2015).

In Lewis and Thompson (2015) a two-stage metaheuristic-based algorithm is proposed for the course timetabling problems. The research analyses the effects of solution space connectivity to propose new neighborhood operators. When the size of the scheduling problem is large, usually the problem becomes intractable. To manage this, two decomposition algorithms are proposed in Al-Yakoob and Sherali (2015). The algorithm comprises a two-stage modeling approach and a mixed-integer programming formulation that attempts to select valid combinations of scheduled from the set of all feasible solutions. A hyper-heuristic is presented in Pillay and Özcan (2019), in which the arithmetic and hierarchical heuristics are combined. The authors use genetic programming, genetic algorithms, and the generation of random heuristic combinations. Genetic programming is used to evolve arithmetic heuristics. The generation of random heuristic combinations is examined for the generation of hierarchical heuristics.

In some research, population-based algorithms are combined with local search algorithms. A hybrid cat swarm optimization algorithm is used in Skoullis et al. (2017) to solve the school timetabling problem. The hybrid algorithm employs the population-based algorithm in conjunction with a local search algorithm. One important criterion in optimization is to find solutions that are robust to input changes. It is specifically true for timetabling problems practically, as there may be some last-minute changes to the availability of teachers, changes in courses, etc. To manage this, a new robustness measure is defined in Akkan and Gulcu (2018), and the optimization process is considered as a multi-objective problem. The authors hybridize GA with SA and the hill-climbing algorithm to solve the problem.

Some research use local search algorithms to solve the problem. In Goh et al. (2017) a tabu search and simulated annealing algorithms are combined into an iterative two-stage procedure. The authors also propose a new neighborhood scheme, which is a new way of estimating local optima and a reheating method. An iterated local search algorithm is proposed in Song et al. (2018), which consists of a simulated annealing and a moderate perturbation procedure. A set of parallel local search algorithms is proposed in Saviniec et al. (2018), which consists of sub-populations. Each sub-population uses a different local search algorithm and some solutions can migrate from one sub-population to another. In Leite et al. (2019), a fast simulated annealing algorithm is proposed for exam timetabling problems, in which ten temperature bins are formed, and each selected is only moved if the exam has accepted moves in the immediately preceding temperature bin. In this method, if an exam has zero accepted movements in the preceding temperature bin, it becomes crystallized and so the number of evaluations is reduced.

The Quantum evolutionary algorithm was designed for the class of combinatorial optimization problems and thus we believe the algorithm is highly suitable for this problem. However, not many works have considered QEA to solve timetabling problems. Among the first attempts in this field are the works presented in Zheng et al. (2009). In YuZheng and Jingfa Liu (2011), a novel quantum evolutionary algorithm is used to solve the heavily constrained university scheduling problem. Quantum evolutionary algorithms are used in Tang and Yang (2013) to solve a bus timetabling problem.

In a set of papers, in Prugel-Bennett and Tayarani-Najaran (2011); Tayarani-N and Prügel-Bennett (2015); Tayarani-N. and Prügel-Bennett (2015); Tayarani-N. and Prugel-Bennett (2014), we studied the fitness landscape of a number of optimization problems. In these papers, we tried to promote the idea that studying the fitness landscape of problems provides useful insights in designing algorithms. In this paper, we decided to take one step further and use the insight that the fitness landscape analysis provides to develop new problem solvers.

Despite the importance of the timetabling problem and understanding its fitness landscape properties, the fitness landscape of this problem has not been widely studied in the literature. Therefore, this paper performs an analysis of the fitness landscape of the problem and study some properties that have not been studied in the literature. Different properties of the fitness landscape are studied in this paper, including the density of states, auto-correlation, correlation length, time to satisfy

the constraints, size of the feasible regions, probability of finding feasible regions, and the distance between the feasible regions.

This paper proposes a novel operator for the quantum evolutionary algorithm in solving the timetabling problem. The analysis performed in the fitness landscape of the problem suggests that there is a correlation between the fitness of local optima and their distance to the best solutions. This suggests that good solutions are clustered in the landscape. Then an algorithm is proposed that explores the search space and collects information about the structure of the landscape. This information is then used to cluster the population and guide the search algorithm towards better regions in the search space.

The rest of this paper is organized as follows. Section 2 describes the timetabling problem. Section 3 performs a fitness landscape analysis on the problem. Quantum evolutionary algorithms and their formulation for solving the timetabling problems are explained in section 4. The proposed algorithm is described in section 5. Section 6 presents the experimental studies and finally section 7 concludes the paper.

2. Timetabling Problem

It has been more than 40 years since the timetabling problem has attracted the attention of computer science researchers. The timetabling problem is a problem which every department of every university, every school, and every sports league or job scheduler faces. Although such widespread, there is no universal form of the problem and the problem is context-dependent which changes even from one semester to another. There is a group of problems that are considered as timetabling problem, including nurse rostering Cheang et al. (2003); Burke et al. (2004), sports timetabling Easton et al. (2004), transportation timetabling Kwan (2004), university course timetabling Schaerf (1999); Burke and Petrovic (2002); Petrovic and Burke (2004), university exam timetabling Eley (2006), employee timetabling Detienne et al. (2009) and train timetabling Caprara et al. (2006). This paper studies the landscape of the high school timetabling problem. The high school timetabling problem is defined as assigning t number of teachers to k number of classes, where each class is assigned to a fixed classroom. There are s different subjects, d number of working days in each week, p number of periods in each day, and the total number of hours that should be taught in each week is n (referred to as the number of time-slots). The assignment of teachers to classes must satisfy a set of hard constraints. There are also some soft constraints that should be taken into account. The set of hard constraints can be described as follows Tayarani-N (2020),

1. H1: A teacher cannot be assigned to more than one class at each time-slot.
2. H2: A class cannot be assigned to more than one teacher at each time-slot.
3. H3: Each teacher has to be assigned to a particular number of time-slots per week.
4. H3': A teacher who teaches subject s, and is assigned to class c has to hold at least h_{cs} of his/her lectures in class c in each weak.
5. H4: Each class must hold a particular number of lectures per week.
6. H5: A teacher can be assigned to a time-slot if he/she is available for that time.
7. H6: A class cannot be free at any of the time-slots, except the last time-slots each day.
8. H7: Not more than one teacher can be assigned to a subject in a given class.

Along with the hard constraints which must be applied to the final solutions, there are some soft constraints that do not necessarily need to be applied, but if applied, fitter solutions are achieved. In solving the problem with QEA, soft constraints can be considered as the cost function f, which distinguishes the solutions. Such constraints are the problem and even teacher dependent. For example, a teacher may prefer to have all of his classes on two consecutive days, or another may want to balance his/her classes throughout the week. For the case of **hdtt** problem ("hdtt" stands for "hard timetabling") , f is the weighted sum of the total penalties incurred by the teachers,

$$f = \sum_{s \in S} \sum_{t \in T_s} \gamma_{ts} z_{ts}, \tag{1}$$

where γ_{ts} is a coefficient representing the importance of the timetable for teacher t, teaching subject s. Hereafter we use ts to refer to the teacher who teaches the subject s. The total penalty incurred by teacher t teaching subject s represented by z_{ts} is calculated as the weighted sum of six penalties:

$$z_{ts} = \sum_{i=1}^{6} \theta_i z_{ts}^i, \tag{2}$$

where z_{ts}^i is the penalty for the i-th criterion and θ_i is the coefficient showing the importance of the corresponding penalty. Here we define each of the six penalties.

1. The first penalty indicates the number of times teacher ts is assigned to teach at a time-slot, that is not his/her desirable time-slot and is found through the following equation,

$$z_{ts}^1 = \sum_{a_{pd}=1} \sum_{c \in C} x_{ctspd}. \tag{3}$$

2. The second penalty counts the number of hours the teachers are idle. This penalty is calculated as follows:

$$z_{ts}^2 = \sum_{d \in D} \left[p_d'' - p_d' - \sum_{p \in P} \sum_{c \in C} x_{ctspd} \right], \tag{4}$$

where p_d' and p_d'' are the first and the last teaching time-slots a teacher teaches on day d respectively.

3. The third penalty counts the number of times teacher ts holds his/her course in class c more than once in a day but in two non-consecutive time-slots. The penalty is found as follows:

$$z^3_{ts} = \sum_{d\in D}\sum_{c\in C}[\![\left(\sum_{p\in P} x_{ctspd} > y_{cts}\right)\wedge$$

$$\left(x_{ctspd} \neq x_{cts(p+1)d}\right)]\!], \quad (5)$$

where the operator $[\![predicate]\!]$ returns '0', if '$predicate = F$' and '1' if '$predicate = T$'.

4. The fourth penalty calculates the number of days that the number of contact hours of teacher st with class c differs from the average daily contact hours for subject s and is found through the following function:

$$z^4_{ts} = \sum_{c\in C}\sum_{d\in D}[\![|\sum_{p\in P} x_{ctspd} - \frac{1}{n_d}h_{cs}y_{cts}| > 1]\!] \quad (6)$$

5. The fifth penalty shows the number of days that the teaching hours of teacher st differs from his/her average number of teaching hours per day. This penalty function is defined as:

$$z^5_{ts} = \sum_{d\in D}[\![|\sum_{c\in X}\sum_{p\in P} x_{ctspd} - \frac{1}{n_d}h_{ts}| > 1]\!] \quad (7)$$

6. The sixth penalty shows the number of times teachers are assigned to teach at the last time-slots of the days and is found as follows:

$$z^6_{ts} = \sum_{d\in D}\sum_{c\in C} x_{ctsn_pd}. \quad (8)$$

3. Landscape Analysis

In this section, some properties of the fitness landscape of the Timetabling problem for the randomly drawn problem instances are studied. We try to show the effect of the size of the problem on different properties of the fitness landscape. The results show how and why the problem becomes harder as the system size grows, and show which parameters have a higher effect on the problem difficulty.

3.1. Phase-Transition

It is shown that as the number of constraints increases, many constraint satisfaction problems exhibit a transition in their difficulty. This is called the phase transition, at which the behavior of the problems changes dramatically, from almost all random instances being satisfiable, to almost all instances being unsatisfiable. Many problems show this behavior, like Maximum Satisfiability problem Monasson et al. (1999); Coppersmith et al. (2004), Graph-Colouring problem Mizuno et al. (2000); Barbosa and Ferreira (2004); Tayarani-N and Prügel-Bennett (2015), Planning problem Rintanen and ludwigs-universitat Freiburg (2004); Zhou and Yin (2010) and Scheduling problems Herroelen and De Reyck (1998); Donati et al. (2008). This section studies this property in the Timetabling problem. To show how the ratio of the solvable problems changes with the number of classes, Figure 1 shows the proportion of solvable problems as a function of the number of classes n_c and the total number of time-slots n. The number of days is $n_d = 5$ and the number of periods in each day is $n_p = 7$. To make this graph, 1000 problem instances are randomly constructed, and using a search algorithm, the constraints are satisfied. Then count the number of problems for which a solution with no unsatisfied hard constraint is found, the proportion of the solvable solutions is calculated. The graph clearly shows that there is a phase transition, with the total number of time-slots. Note that the total number of time-slots, n, here is the total number of hours that should be taught in the school. On the other hand, the maximum number of hours a school can hold is $n_c \times n_d \times n_p$, where n_c is the number of classes, n_d is the number of days, and n_p is the number of periods in each day. Obviously, the total number of time-slots, n, must always be less than the maximum capacity of the school. As the number of time-slots gets closer to the maximum number of hours the school can hold, the proportion of the solvable instances decreases until it becomes zero at the maximum number of hours.

For example, for the case of $n_c = 6$, the number of hours the school can hold is $6 \times 5 \times 7 = 210$. Therefore, the maximum number of time-slots can be 210. It is clear that for the case of $n_c = 6$, any value of time-slots, n, above 210 results in problems being unsolvable. For $n = 210$, only a fraction of problems is solvable. This is because although the number of time-slots equals the number of hours the school can hold, because of the constraints like teachers' availability, some problems can be unsolvable.

Regarding the comment made about the number of time-slots 350, the referee should look at the problem from the opposite. Note that the number of time-slots is the number of slots that we need to teach at the school. But the total number of slots that the school can hold (the school capacity) is 300. Obviously, it is not possible to fit 350 slots in a school with a capacity of 300. Here the question is why when the school has the capacity of 350 slots per week, we can only insert 300 hours of teaching? The answer is that although there are still 50 slots empty at the school, the conflict between teacher's availability and other constraints make the problem unsolvable. Therefore 50 hours/class of the school go to waste.

The same experiment is performed for a different numbers of teachers. The number of teachers does not change the location of the phase of the transition, but it changes the sharpness. It is shown in Figure 2, where the proportion of solvable problems is presented for a different numbers of teachers. Note that all the curves drop to zero at number of time-slots equal to $n_c \times n_d \times n_p = 210$. It is clear that the smaller the number of teachers, the faster the proportion of the solvable problems drops. It is obvious why this happens, the more teachers available, the more robust the problem will be, so getting closer to the maximum number of possible

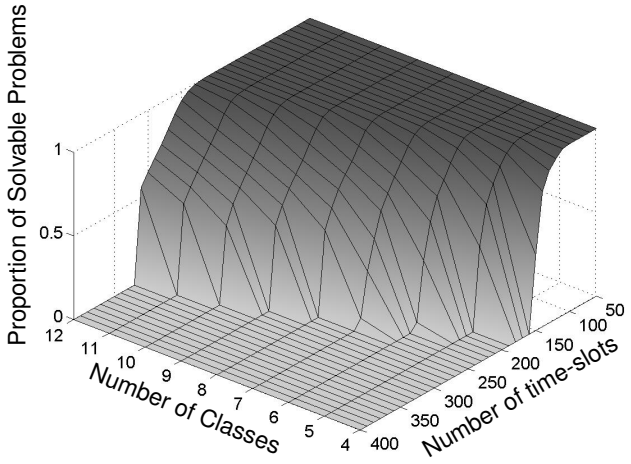

Figure 1: The proportion of solvable problems as a function of number of classes n_c, and number of time-slots n, for number of teachers $n_t = 16$. The data are averaged over 1000 random problem instances.

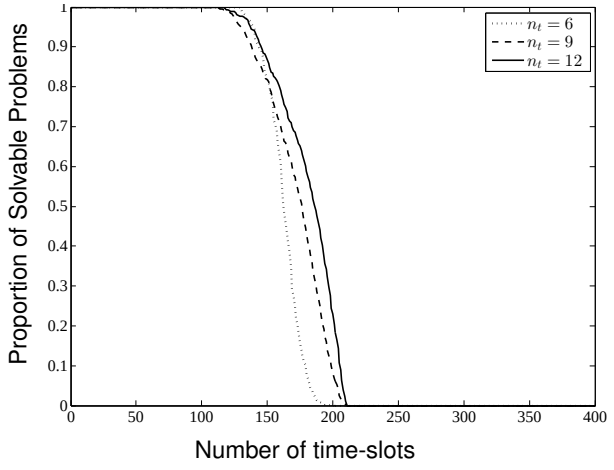

Figure 2: The proportion of solvable problems as a function of number of time-slots n for number of teachers $n_t = 6, 9$ and 12 for number of classes $n_c = 6$. The data are averaged over 1000 random problem instances.

time-slots ($n_c \times n_d \times n_p$), there still could be ways of satisfying the hard constraints.

3.2. Density of States

Our study begins by studying the statistical properties of the randomly drawn configurations in the landscape. These properties show the general behavior of random solutions in the landscape. The density of states shows the number of configurations at each cost level. By random sampling of solutions and finding the histogram of the cost of the random solutions, the spread of costs around the mean can be computed. The fitness landscape of the timetabling problem consists of two main regions, the region in which the hard constraints are not satisfied, and the cost of the solutions is the number of unsatisfied hard constraints. The other region is the region in which the hard constraints are satisfied, so the cost of the solutions is found based on the soft constraints.

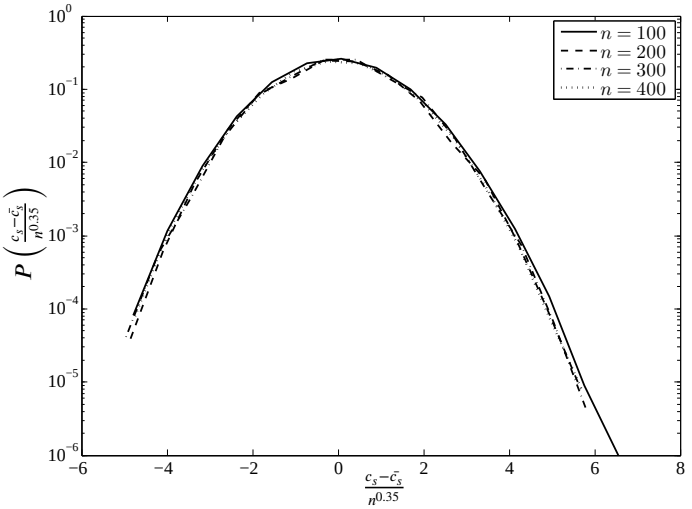

Figure 3: Histogram of hard-costs on a logarithmic scale for random solutions in particular instances for different number of time-slots. The number of classes $n_c = 6$ and number of teachers is $n_t = 16$. The horizontal axis is scaled by $n^{0.35}$. The results are for randomly constructed problem instances for 10^7 sampling.

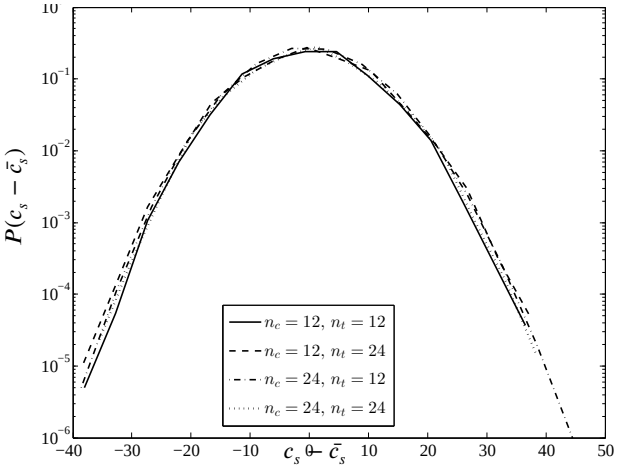

Figure 4: Histogram of hard-costs on a logarithmic scale for random solutions in particular instances for different number of classes and teachers. The number of time-slots is $n = 200$. The results are for randomly constructed problem instances for 10^7 sampling.

First, the density of states for the hard cost of the solutions is studied. The distribution of the costs around the mean for different number of time-slots n, for number of classes $n_c = 6$ and number of teachers $n_t = 16$ is shown in Figure 3. The data are rescaled by $n^{0.35}$, so the best match for a different number of time-slots is found. It seems that the standard deviation of the distribution grows approximately $n^{0.35}$.

Figure 4 shows the histogram for different number of classes and teachers. It seems that the variance does not change with the number of classes or the number of teachers.

Figure 5 shows the logarithm of the histogram of the

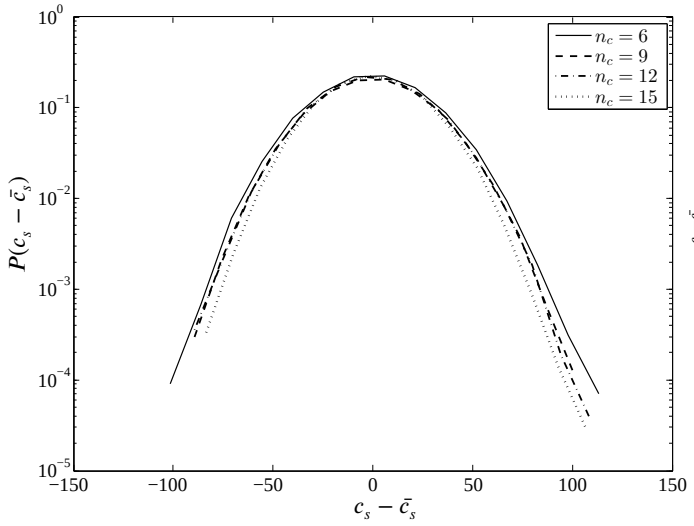

Figure 5: Histogram of soft-costs on a logarithmic scale for random solutions in particular instances for different number of classes. The number of teachers is $n_t = 16$ and the number of time-slots is $n = 100$. The results are for randomly constructed problem instances for 10^7 sampling.

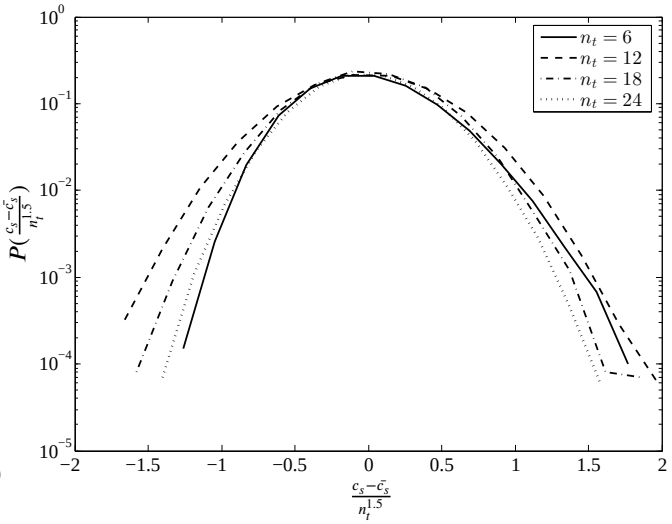

Figure 6: Histogram of soft-costs on a logarithmic scale for random solutions in particular instances for different number of teachers scaled by $n_t^{1.5}$. The number of classes is $n_c = 6$ and the number of time-slots is $n = 100$. The results are for randomly constructed problem instances for 10^7 sampling.

soft-costs around the average cost for different number of classes for number of teachers $n_t = 16$, and number of time-slots $n = 100$. Regardless of the number of classes, the shape of the distribution is quite similar for each randomly drawn problem instance and the costs of the random configurations are approximately normally distributed around the average.

The same experiment is performed for a different numbers of teachers, and the logarithm of the histogram scaled by $n_t^{1.5}$ is represented in Figure 6, where the number of classes is $n_c = 6$, the number of time-slots is $n = 100$ and the number of sampling is 10^7. Although the curves do not clearly lay on top of each other, all of them show a normal shape distribution around the mean.

3.3. Auto-Correlation

Auto-correlation measures the local ruggedness of the fitness landscape by measuring the expected correlation of a random configuration and that of a configuration where τ randomly chosen steps Weinberger (1990) are taken. The auto-correlation of the landscape is given by Weinberger (1990),

$$R(\tau) = \frac{1}{\sigma^2} E\left(\left(f(t+\tau) - \bar{f}\right)\left(f(t) - \bar{f}\right)\right), \qquad (9)$$

where $f(t)$ is cost at step t and σ^2 is the variance in the cost for random configurations of the problem instance. This measures the expected changes in the fitness of the function as τ moves have taken. We have computed the auto-correlation for different number of classes, teachers and time-slots. The auto-correlation function appears to fall off approximately exponentially as

$$R(\tau) \approx e^{-\tau/l}, \qquad (10)$$

where l is known as the correlation length Stadler (1996). Figure 7 shows the correlation length of the landscape of hard constraints versus the number of classes on a log-log scale. The data show that as the number of classes grows, the correlation length grows as $l \approx n_c^{0.16}$, which means that as the number of classes grows, the neighbor solutions in the landscape become more correlated. The higher correlation length is usually interpreted as an indicator for a less rugged landscape, and therefore an easier problem. This is obviously not the case for the Timetabling problem, as more number of classes mean a much harder problem. The same experiment is performed for a different number of teachers, and the data show similar behavior, the correlation length grows as $l \approx e^{0.24 n_t}$. There is also a relationship between the correlation length and the number of time-slots. The data suggest that the correlation length grows linearly with the number of time-slots, for $n_c = 6$, and $n_t = 16$ it grows as $l \approx 0.33 n_p$.

3.4. Landscape of Hard Constraints

The landscape of the timetabling problem consists of two fitness space, the hard constraints, and the soft constraints. A search algorithm has to search in the hard constraint landscape to find the region in which all the hard constraints are satisfied, where we call the feasible region. After reaching the feasible region, the algorithm starts the second phase of its search process, at which it tries to minimize the soft constraint cost. In this section, we investigate the properties of the feasible regions.

3.4.1. Time to feasible regions

We start our analysis by studying the time it takes for a local-search algorithm to reach a feasible region, starting from a random configuration. The local search algorithm checks all the neighbors of the configuration and moves to the neighbor with the best fitness. The process is performed until a feasible region is found. Figure 8 shows the relationship

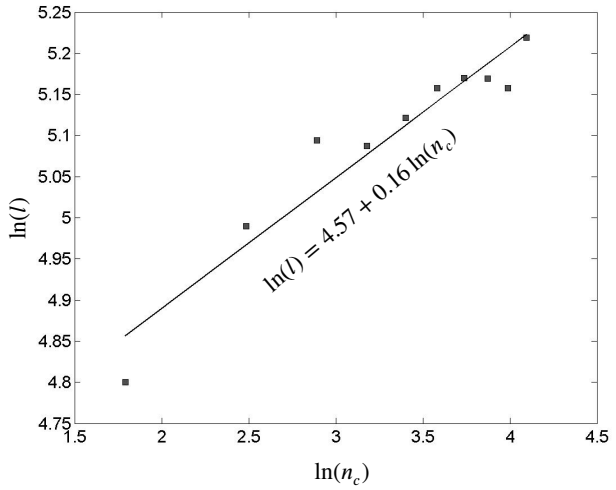

Figure 7: Correlation length of the landscape versus the number of classes on a log-log scale. This is for number of teachers $n_t = 16$, number of time-slots $n = 200$. The data are averaged over 100 problems, 10^6 walks on each.

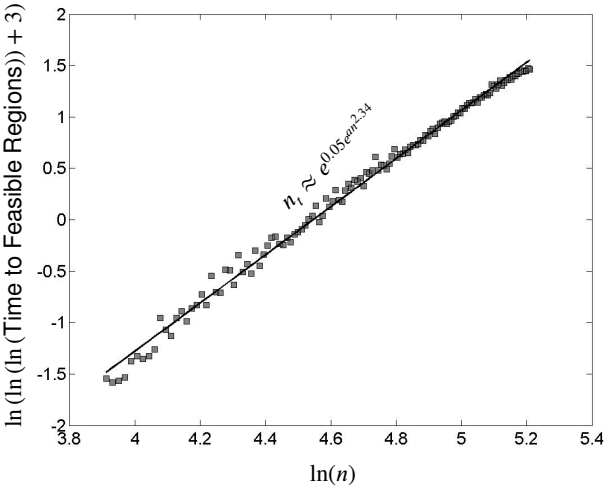

Figure 8: Time to satisfy the hard constraints versus the number of time-slots. This is for number of classes $n_c = 12$ and number of teachers $n_t = 16$. The data are averaged over 1000 problem instances and 10^5 descents on each.

between the number of time-slots, n, and the time it takes for the local search algorithm to satisfy the hard constraints. Under the scaling used in this figure, the data fit a straight line. Although the complex scaling of the figure is unlikely to model the true behavior of the timetabling problem, it is indicative of the complexity of the landscape of the problem. The fitting line in Figure 8 suggests that the time it takes to reach a feasible region increases as $t \approx e^{0.05e^{an}}^{2.34}$, where $a = e^{-10.64}$. This means that as the number of time-slots gets closer to the phase transition, the time to feasible regions grows much faster than exponentially. This is an interesting finding as the analysis suggests that the correlation length grows linearly with the number of time-slots. This suggests that although the fitness landscape becomes less rugged as the number of timeslots grows, it takes a longer time to solve the problem. This seems counter-intuitive as the fitness ruggedness is interpreted as problem hardness. This can be explained by the fact that it is not only the landscape ruggedness that determines the problem hardness. In some problems, it is the existence of large plateau regions that make the problem hard for finding better solutions. A large plateau can make the fitness landscape less rugged. Note that the increase in time to satisfy the hard constraints only occurs close to the phase transition, and below the phase transition this rapid growth rate is not observed.

The data presented in Figure 8 show that by increasing the number of constraints (time-slots) the time it takes to satisfy the hard constraints grows rapidly. The time also depends on the number of classes, as this number is another constraint on the problem. Figure 9 shows the relationship between the time to satisfy the hard constraints and the number of classes. This constraint also has a significant effect on the time to feasible regions, where the time grows much faster than exponentially as the number of classes decreases (the number of constraints increases).

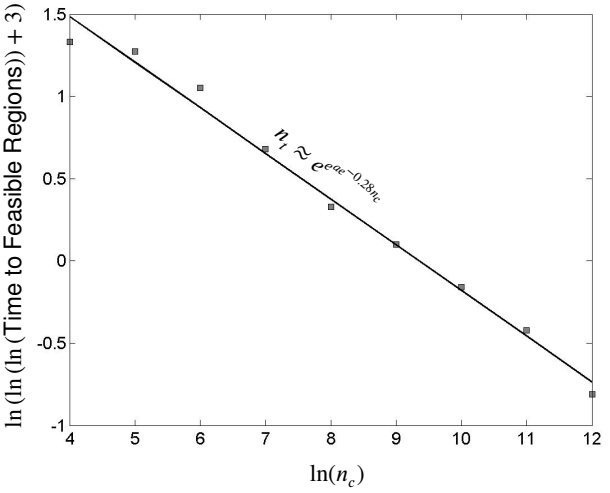

Figure 9: Time to satisfy the hard constraints versus the number of classes. This is for number of time-slots $n = 120$ and number of teachers $n_t = 16$. The data are averaged over 1000 problem instances and 10^5 descents on each.

3.4.2. Size of the feasible regions

The other important property of the feasible regions is their size, as increasing the number of constraints leads to a decrease in the size of the feasible regions. This section studies the effect of the number of time-slots on the size of the feasible regions.

We saw in Figure 1 that as the number of time-slots increases, we approach a phase transition, at which problems become unsolvable. This means that increasing the number of time-slots decreases the size of the feasible regions until the feasible regions disappear. In order to show this relationship, Figure 10 shows the natural logarithm of the natural logarithm of the size of the feasible regions, versus the natural logarithm of

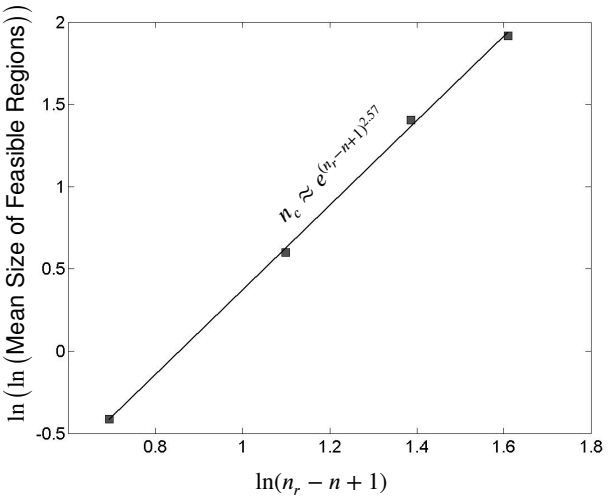

Figure 10: Natural logarithm of the natural logarithm of the number of configurations on the feasible regions, n_c, versus the natural logarithm of the number of time-slots at the phase-transition, minus the number of time-slots plus one. Where n_r is the number of time-slots at the phase-transition and n is the total number of time-slots. This is for number of classes $n_c = 4$, number of teachers $n_t = 4$, number of days $n_d = 3$ and number of periods is each day is $n_p = 3$. This is averaged over 1000 problem instances for 10^5 descends on each.

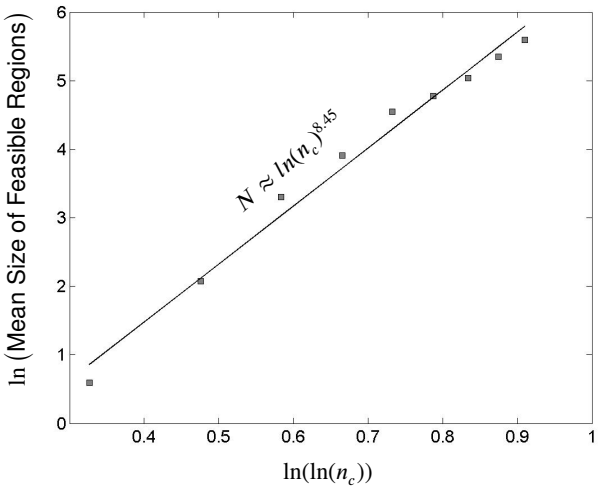

Figure 11: Natural logarithm of the number of configurations on the feasible regions, N, versus the natural logarithm of the natural logarithm of the number of classes. This is for number of time-slots $n = 23$, number of days $n_d = 3$ and number of periods in each day $n_p = 3$. This is averaged over 1000 problem instances for 10^5 descends on each.

the number of time-slots at the phase-transition minus the number of time-slots plus one. The data suggest that the size of the feasible regions decays as $n_c \approx e^{(n_r-n+1)^{2.57}}$ with the number of time-slots. This means that increasing the number of time-slots decreases the size of feasible regions faster than exponentially, and thus, the number of possible solutions decays very fast as the number of time-slots grows.

Figure 11 shows the relationship between the number of classes and the size of the feasible regions. On the scaling used in this figure, the data almost fit a straight line, which suggests that the size of the feasible regions grows as $N \approx ln(n_c)^{8.45}$.

3.5. Finding the feasible regions

One important property of the feasible regions is the probability of the local search algorithms finding them. Figure 12 shows the probability of finding the feasible regions versus the size of the feasible regions for a particular problem instance with number of classes, $n_c = 5$, number of time-slots, $n = 24$ and number of teachers, $n_t = 5$. The data show that there is no particular relationship between the size of the feasible regions and the probability of finding them. This means that there is no correlation between the size of the feasible regions and the size of the basin of attraction of them.

In the timetabling problem, at the first phase a local search algorithm finds a feasible region, then considering the soft constraints as the objective, another local search algorithm explores the feasible region in search of a good solution. This property makes the probability of finding solutions depend on the probability of finding the feasible region in which

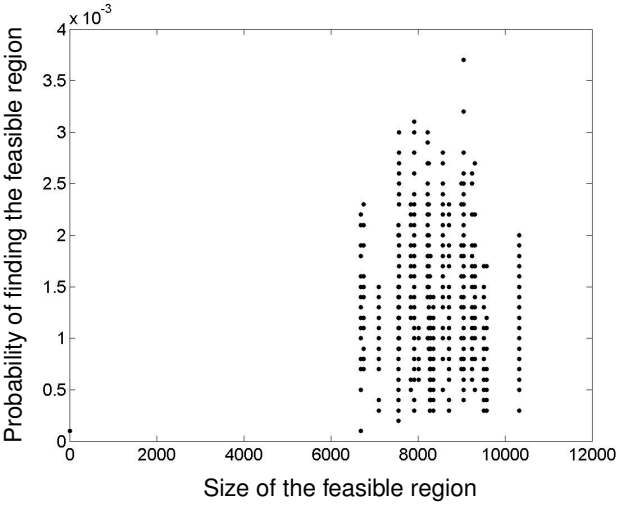

Figure 12: Probability of finding the feasible regions versus the size of the feasible regions. This is for a particular instance for number of classes, $n_c = 5$, number of time-slots, $n = 24$ and number of teachers, $n_t = 5$. The number of descents is 10^6.

the solution lies. The reason for such behavior is that there is no correlation between the hard landscape and the soft landscape. Figure 13 shows the probability of finding feasible regions against the soft cost of the best local optimum in the feasible region. The data show no correlation between the probability of finding a feasible region and the cost of the best solution in the region. This property is clearly an undesirable one because when searching for a feasible region, the local search algorithms do not (cannot) consider the soft cost, so with a higher probability, they end up in a feasible region with a bigger basin of attraction, which is not necessarily

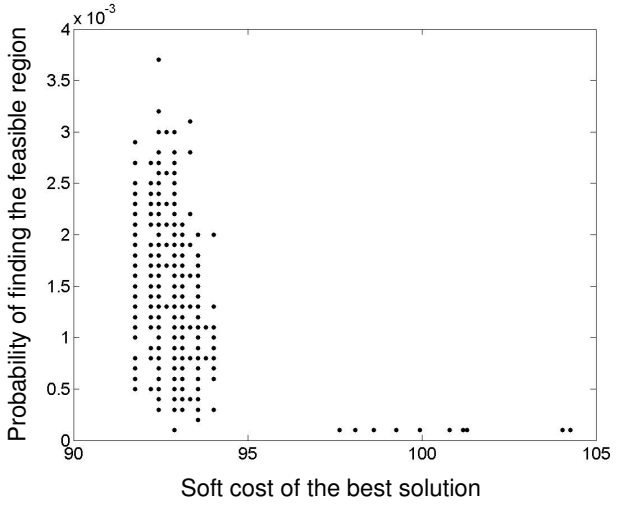

Figure 13: Probability of finding the feasible regions versus the soft cost of the best solution on the feasible region. This is for a particular instance for number of classes, $n_c = 5$, number of time-slots, $n = 24$ and number of teachers, $n_t = 5$. The number of descents is 10^6.

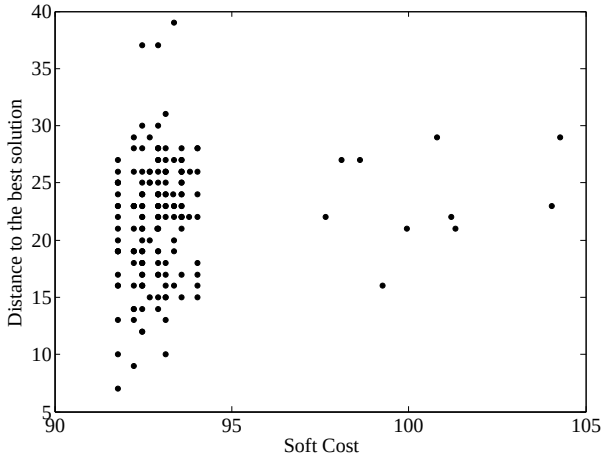

Figure 14: The distance between the best solutions in each feasible region to the best observed solution. This is for a particular instance for number of classes, $n_c = 5$, number of time-slots, $n = 24$ and number of teachers, $n_t = 5$. The number of descents is 10^6.

the region containing a solution with a desirable soft cost.

In order to find the distribution of good solutions in the landscape, we analyze the distance between the good solutions. The distance between the two solutions is simply calculated as the Hamming distance between two solutions. Figure 14 shows the distance between the best solutions in each feasible region to the best-observed solution versus the soft cost of the solutions. The data suggest that good solutions tend to be closer to the best solution, even though they are located in different feasible regions. This is an important property of the fitness landscape that can be exploited when developing optimization algorithms.

4. Quantum Evolutionary Algorithms

QEA uses a novel representation based on the aforementioned concept of q-bits Tayarani-N and Akbarzadeh-T (2014). Consider i-th individual in τ-th generation defined as an n-qubit as Han and Kim (2002)

$$\begin{bmatrix} \alpha_{i1}^{\tau} & \alpha_{i2}^{\tau} & \dots & \alpha_{ij}^{\tau} & \dots & \alpha_{in}^{\tau} \\ \beta_{i1}^{\tau} & \beta_{i2}^{\tau} & \dots & \beta_{ij}^{\tau} & \dots & \beta_{in}^{\tau} \end{bmatrix}, \tag{11}$$

where $\left|\alpha_{ij}^{\tau}\right|^2 + \left|\beta_{ij}^{\tau}\right|^2 = 1$, $j = 1, 2, ..., n$, where n is the number of q-bits, i.e., the string length of the q-bit individual, $i = 1, 2, ..., m$, where m is the number of possible solutions in the population and τ is generation number of the evolution. Since a q-bit is a probabilistic representation, any superposition of states is simultaneously represented. If there is, for instance, a three-q-bits ($n = 3$) individual such as

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{\sqrt{2}}{\sqrt{3}} & \frac{\sqrt{3}}{2} \end{bmatrix}, \tag{12}$$

alternatively, the possible states of the individual can be represented as

$$q_i^{\tau} = \frac{1}{2\sqrt{6}}|000\rangle + \frac{1}{2\sqrt{2}}|001\rangle + \frac{1}{2\sqrt{3}}|010\rangle + \frac{1}{2}|011\rangle + \frac{1}{2\sqrt{6}}|100\rangle + \frac{1}{2\sqrt{2}}|101\rangle + \frac{1}{2\sqrt{3}}|110\rangle + \frac{1}{2}|111\rangle . \tag{13}$$

Note that the square of above numbers are true probabilities, i. e., the above result means that the probabilities to represent the states $|000\rangle$, $|001\rangle$, $|100\rangle$ and $|010\rangle$ are $1/24$, $1/8$, $1/24$ and $1/12$ respectively. Consequently, the three-q-bits system of (equation 12) could carry all eight states information at the same time.

Evolutionary computing with the q-bit representation has a better characteristic of diversity than classical approaches since it can represent a superposition of states. Only one q-bit individual such as equation 13 is enough to represent eight states, whereas, in classical representation, eight individuals are needed. Additionally, along with the convergence of the quantum individuals, the diversity gradually fades away and the algorithm converges.

4.1. Quantum Gates Assignment

The common mutation is a random disturbance of each individual, promoting exploration while also slowing convergence. Here, the quantum bit representation can be simply interpreted as a biased mutation operator. Therefore, the current best individual can be used to steer the direction of this mutation operator, which will speed up the convergence. The evolutionary process of a quantum individual is completed through the step of "update $Q(\tau)$". A quantum rotation gate is described below. Specifically, a q-bit individual q_i^{τ} is updated using the rotation gate $U(\theta)$ in this algorithm. The k-th q-bit of the i-th quantum individual generation τ, $[\alpha_{ik}^{\tau} \quad \beta_{ik}^{\tau}]^T$ is updated

Table 1
Lookup Table of $\Delta\theta$, the rotation gate. x_i is the i-th bit of the observed binary solution and b_i is the i-th bit of the best found binary solution.

x_i	b_i	$f(x) \geq f(b)$	$\Delta\theta$
0	0	false	0
0	0	true	0
0	1	false	0.01π
0	1	true	0
1	0	false	-0.01π
1	0	true	0
1	1	false	0
1	1	true	0

as Han and Kim (2002),

$$\begin{bmatrix}\alpha_{ik}^{\tau+1}\\ \beta_{ik}^{\tau+1}\end{bmatrix} = \begin{bmatrix}\cos(\Delta\theta) & -\sin(\Delta\theta)\\ \sin(\Delta\theta) & \cos(\Delta\theta)\end{bmatrix}\begin{bmatrix}\alpha_{ik}^{\tau}\\ \beta_{ik}^{\tau}\end{bmatrix}, \tag{14}$$

where $\Delta\theta$ is the rotation angle and controls the speed of convergence and is determined from Table 4.1. Han and Kim (2002) show that these values for $\Delta\theta$ have better performance.

This section describes the way QEA is adopted for timetabling problems. In order to be able to solve the problem with evolutionary algorithms, at the first step, a representation of the possible solutions has to be proposed. In this paper, we use binary representation. Each binary solution is represented with a five-dimensional binary array x_{ctspd}, where,

$c \in C = \{1, 2, ..., n_c\}$ represents the c-th class and n_c is the total number of classes.

$s \in S = \{1, 2, ..., n_s\}$ represents the s-th subject and n_s is the total number of subjects.

$t \in T_s = \{1, 2, ..., n_{t,s}\}$ represents the t-th teacher teaching subject s and $n_{t,s}$ is the total number of teachers who teach the s-th subject.

$p \in P = \{1, 2, ..., n_p\}$ represents the p-th period(time-slot) and n_p is the total number of time-slots in each day.

$d \in D = \{1, 2, ..., n_d\}$ represents the d-th day in the week, and n_d is the total number of working days in each week.

If x_{ctspd}=1, it means that in c-th class, t-th teacher will teach s-th subject in p-th time-slot of d-th day of the week. But according to the constraints, not every x can represent a feasible solution. The constraints have to also be considered. In order to check the constraints, two other variables are introduced. A three dimensional array y, where y_{cts}=1 if the t-th teacher is scheduled to teach s-th subject in c-th class. And a four-dimensional array a where a_{tspd}=1 if the t-th teacher, teaching s-th subject is available at p-th time-slot of d-th day. Using such variables and notations the constraints can be formulated as follows,

$$\text{H1: } \sum_{c\in C} x_{ctspd} \leq 1,$$

$$\text{H2: } \sum_{s\in S}\sum_{t\in T_s} x_{ctspd} \leq 1,$$

$$\text{H3: } \sum_{c\in C}\sum_{d\in D}\sum_{p\in P} x_{ctspd} = h_{ts},$$

$$\text{H3': } \sum_{d\in D}\sum_{p\in P} x_{ctspd} \geq h_{cs},$$

$$\text{H4: } \sum_{s\in S}\sum_{t\in T_s}\sum_{d\in D}\sum_{p\in P} x_{ctspd} = h_c,$$

$$\text{H5: } \sum_{c\in C} x_{ctspd} \leq 0, \quad \text{if } a_{ts} = 0,$$

$$\text{H6: } \sum_{s\in S}\sum_{t\in T_s} x_{cst(p+1)d} - \sum_{s\in S}\sum_{t\in T_s} x_{ctspd} \geq 0, \quad p = 1, ..., n_p - 2,$$

$$\text{H7: } \sum_{t\in T_s} y_{cts} = 1,$$

$$\text{H8: } y_{cts} \geq x_{ctspd}, \tag{15}$$

where $t \in T_s$, $s \in S$, $p \in P$, $d \in D$, $c \in C$. H7 means that no more than one teacher is assigned to a subject in a given class and, H8 means that only the specific teacher in H7 is assigned to the h_{cs} lectures of class c for subject s.

The proposed representation method has the flexibility to be applied to every scheduling problem, the only change needed is to define the constraints in a proper way.

4.2. Using QEA to Solve Timetabling Problem

In this section QEA is modified to adapt it to timetabling problem. The QEA is modified as follows.

Algorithm 1 QEA for timetabling problem

Proposed Algorithm
begin
  $\tau = 0$
1. Initialize Q^0
2. Make X^0 by observing the states of Q^0.
3. Perform local search on X^0 to satisfy hard constraints.
4. Evaluate X^0
5. Store X^0 into B^0. Store the best solutions among X^0 into b.
6. while not termination condition do
  begin
    $\tau = \tau + 1$
7. Make X^τ by observing the states of $Q^{\tau-1}$
8. Perform local search on X^τ to satisfy hard constraints.
9. Evaluate X^τ
10. Update Q^τ using Q-gate
11. if migration-condition then perform migration.
  end
end

QEA has a population of quantum individuals $Q^t = \{q_1^\tau, q_2^\tau, ..., q_m^\tau\}$ where τ is the generation step and m is the size of the population. The QEA procedure in algorithm 1 is described as follows.

1- In our representation, each possible solution is a 5-dimensional solution x_{ctspd}, so each q-individual is a five dimensional array of q-bits. In the initialization step all qubits $\alpha^0_{i,ctspd}$ and $\beta^0_{i,ctspd}$ are initialized with $1/\sqrt{2}$, where m is the size of the population, $i = 1, 2, ..., n$, $c \in C$, $t \in T$, $s \in S$, $p \in P$, $d \in D$. It means that the probability of observing 0 and 1 for all qubits is equal.

2- In this step, the binary solutions $X^0 = \{x_1^0, x_2^0, ..., x_m^0\}$ at generation $\tau = 0$ are created by observing Q^0. In original version of QEA, observing the binary solution $x^\tau_{i,ctspd}$ from qubit $\begin{bmatrix}\alpha^\tau_{i,ctspd} & \beta^\tau_{i,ctspd}\end{bmatrix}$ is performed as below:

$$x^\tau_{i,ctspd} = \begin{cases} 0 & if \;\; U(0,1) < |\alpha^\tau_{i,ctspd}|^2 \\ 1 & otherwise \end{cases}, \quad (16)$$

where $U(.,.)$, is a uniform random number generator.

3- In some problems like Max-Sat and numerical optimization problems, every binary string is a possible solution to the problem. In the constraint satisfaction problems like Knapsack and timetabling problems, there are some hard constraints that must be satisfied. In such problems, not every binary configuration represents a feasible solution. Since evolutionary algorithms are random algorithms and may generate every possible solution in the search space, there should be a method of preventing the algorithm from making infeasible solutions. One way of doing this is by designing a repair function which makes an infeasible solution a feasible one. The problem with the repair function is that there may be many constraints being violated and after repairing a solution, although the solutions become feasible, due to the numerous changes to the solutions, the cost of the solutions is not usually good. In order to solve this problem, a new observation operator is proposed in this paper. In our proposed method, a local search algorithm is performed on the solution to change the solution until it reaches a state, at which all constraints are satisfied. Each step of the proposed local search algorithm is as follows. First, one of the subjects is chosen randomly:

$$s' = \lceil U(0, n_s) \rceil . \quad (17)$$

Then one of the teachers who teaches the subject s is chosen randomly and is put in one of the randomly chosen slots,

$$t' = \lceil U(0, n_t) \rceil , \;\; p' = \left\lceil U(0, n_p) \right\rceil , \;\; d' = \lceil U(0, n_d) \rceil . \quad (18)$$

At the next step, using the availability array, $a_{t'p'd'}$, it is determined if the teacher is available in that time-slot. If the chosen teacher is available (i.e. $a_{t'p'd'}$=1) at the time-slot one of the classes is chosen randomly,

$$c' = \lceil U(0, n_c) \rceil . \quad (19)$$

Finally, the availability of the randomly chosen class is checked. A class is available if

$$\sum_{s \in S} \sum_{t \in T_s} x_{ct'sp'd'} \leq 1, \quad (20)$$

where t' is the randomly chosen teacher, p' and d' show the randomly chosen time-slot. If the class were available, the observation operator is performed to make $x_{i,ctspd}$ from $\alpha_{i,ctspd}$. This process is performed until a binary solution that satisfies all the constraints is achieved.

4- All solutions in X^τ are evaluated using the soft cost function.

5- Store X^0 into B^0. Select the best solution among X^0 and store it to b.

6- The while loop runs until the termination condition is satisfied. The termination condition can be considered as maximum generation condition, convergence condition or the desired fitness is achieved.

7- Observe X^τ from $Q^{\tau-1}$.

8- Perform the local search as described in step 3 to satisfy the hard constraints.

9- Evaluate X^τ using soft cost function.

10- Update Q^τ.

11- A migration is performed in this step.

This is the original version of QEA, that has been modified to adapt to the timetabling problem. To further improve the performance, a new operator is proposed in the next sections.

5. Exploiting Fitness Landscape Properties

To design successful optimization algorithms requires an understanding of the underlying properties of the optimization problems. One way of understanding the properties of the problems is fitness landscape analysis Prugel-Bennett and Tayarani-Najaran (2011); Tayarani-N and Prügel-Bennett (2015); Tayarani-N. and Prügel-Bennett (2015); Tayarani-N. and Prugel-Bennett (2014); Tayarani-N. and Prugel-Bennett (2016). By studying the fitness landscape properties of the timetabling problem, we try to develop new operators for QEA to improve its performance in solving the problem.

It is argued that discovering the properties of the fitness landscape and its structure can help evolutionary algorithms to perform a more informed search in the landscape Tayarani-N. and Prugel-Bennett (2014). However, it is crucial to ask how the information about the fitness landscape can be collected and how the information should be processed to build constructive knowledge and guide the search process. During the search process, evolutionary algorithms search through the search space and visit many solutions and local optima. The visited solutions and their fitness carry information about the backbone structure of the fitness landscape. If exploited properly, the information can be used to inform evolutionary algorithms to search for better parts of the search space for better solutions.

In this paper, a new version of the quantum evolutionary algorithm is proposed for the timetabling problem. In the proposed method, an operator is developed that collects information about the fitness landscape and tries to capture the large scale structure of the fitness landscape. Our proposed algorithm has two parts. One is the quantum evolutionary algorithm that works as a global search and one is a local search.

As shown in Figure 14, good solutions tend to be closer to the global optimum. This suggests that good solutions are

clustered in the landscape and the location of good solutions provides information about where it is more likely to find the global optimum. In order to exploit the information, an operator is proposed that collects information about the best solutions and uses the information to guide the search process towards better regions in the search space. The pseudo-code of the proposed QEA for the timetabling problem is introduced in algorithm 2.

In QEA by converging the algorithm, the quantum bits converge to the true values of $[\alpha \ \ \beta]^T = [0 \ \ 1]^T$ or $[\alpha \ \ \beta]^T = [1 \ \ 0]^T$. In this condition, the algorithm is trapped in a local optimum of which it has little chance of escaping. This weakness is targeted in this paper to improve the performance of the algorithm by detecting the converged q-individuals and guiding them towards better regions in the search space. In the proposed algorithm in each generation, the convergence of the population is calculated and when the population converges, it is reinitialized based on the information it has gathered about the fitness landscape during its previous search steps. The convergence of the population is calculated as Tayarani, N and Akbarzadeh. T (2008),

$$\gamma = \frac{1}{n_c n_t n_s n_p n_d} \sum_{c\in C} \sum_{t\in T_s} \sum_{s\in S} \sum_{p\in P} \sum_{d\in D} \left| 1 - 2 \left| \alpha_{i,ctspd} \right|^2 \right|, \tag{21}$$

where γ is the convergence of the population and m is the size of the population (the number of q-individuals in the population). The convergence here is the distance of the q-individual to the converged state. One way of helping the algorithm escape from the optima is to initialize the population, where it gives the q-individuals a new chance to search and find new solutions. In our proposed algorithm, a database of best-observed solutions is generated and the history of the search process during the past generations is used to make better q-individuals that represent the better parts of the search space with higher probability. Once the q-individuals are trapped in a local optimum, using the collected information, the proposed method reinitializes the q-individuals. The pseudo-code of the proposed algorithm is presented in Algorithm 2.

The proposed method which is described in algorithm 2 and in Figure 15 and 16 has two phases, one is the exploration step, where the algorithm performs a search to discover the landscape and one is the search process, in which the algorithm searches through space. In step 4 of the algorithm, the best solutions found in the local search process are stored in $\mathcal{H}$. Here the set $\mathcal{H}$ contains information about the fitness landscape. In step 9 of the algorithm 2, the proposed landscape estimating local search is performed (algorithm 3).

The proposed local search algorithm tries to explore the fitness landscape and collect data about the location of feasible regions. In order to give the algorithm more exploration and increase its chance of finding as many feasible regions as it can, during its search process, the algorithm uses a tabu mechanism and tries to avoid the feasible regions it has already discovered. Therefore, instead of a random step, only the steps get accepted that move away from the already

Algorithm 2 The proposed algorithm

begin
$\tau = 0$
1. Initialize Q^0
2. Make X^0 by observing the states of Q^0.
3. Perform local search on X^0.
4. Add X^0 to the set $\mathcal{H}$.
5. Evaluate X^0
6. Store X^0 into B^0. Store the best solutions among X^0 into b.

Phase 1: Landscape Estimation

7. while not termination condition do
begin
$\tau = \tau + 1$
8. Make X^τ by observing the states of $Q^{\tau-1}$
9. Perform **landscape estimating search** (algorithm 3) on all $x \in X^\tau$.
10. Evaluate X^τ
11. Update Q^τ using Q-gate
12. Store the best solutions among $B^{\tau-1}$ and X^τ into B^τ
13. Update $\mathcal{H}$.
14. if migration-condition
15. perform migration.
end

Phase 2: Landscape Exploitation

16. while not termination condition do
begin
$\tau = \tau + 1$
17. Make X^τ by observing the states of $Q^{\tau-1}$
18. Perform local search on all $x \in X^\tau$.
19. Evaluate X^τ
20. Update Q^τ using Q-gate
21. Store the best solutions among $B^{\tau-1}$ and X^τ into B^τ
22. Update $\mathcal{H}$.
23. if $\gamma > \rho$ then
24. Cluster the solutions in $\mathcal{H}$ and find κ_i, $i = 1...\kappa$
25. reinitialize q-individuals based on κ_i.
end
end

Algorithm 3 Landscape Estimating Search

while x does not satisfy all constraints
begin
1. for all possible improving moves on x find x' that satisfies $\forall y \in \mathcal{H}, \Delta(y, x) < \Delta(y, x')$.
2. if no solution was found in step 1, find one that satisfies $\forall y \in \mathcal{H}, \Delta(y, x) <= \Delta(y, x')$.
3. if no solution was found in steps 1 and 2, chose one move randomly.
4. x=x'.
end
if x satisfies all constraits
5. perform a local search to optimize soft constraints.

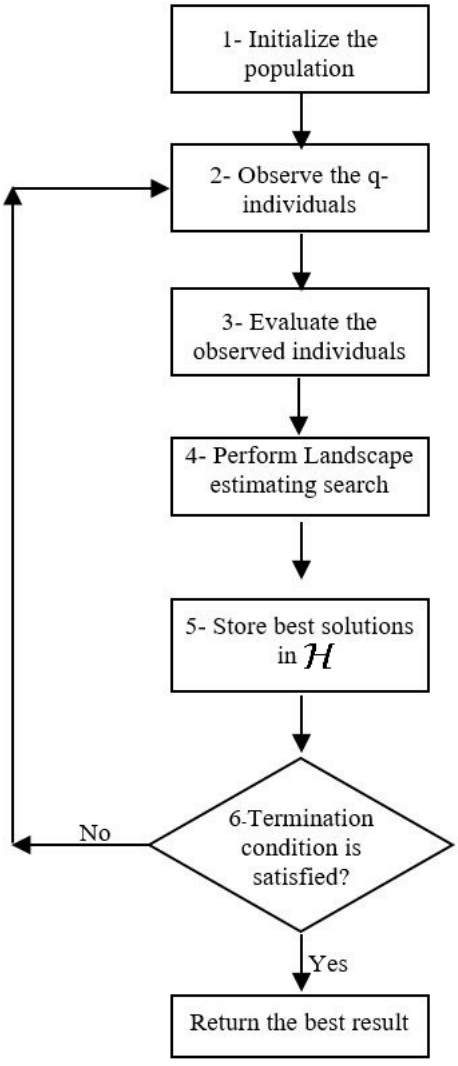

Figure 15: The flowchart of the phase one of the algorithm.

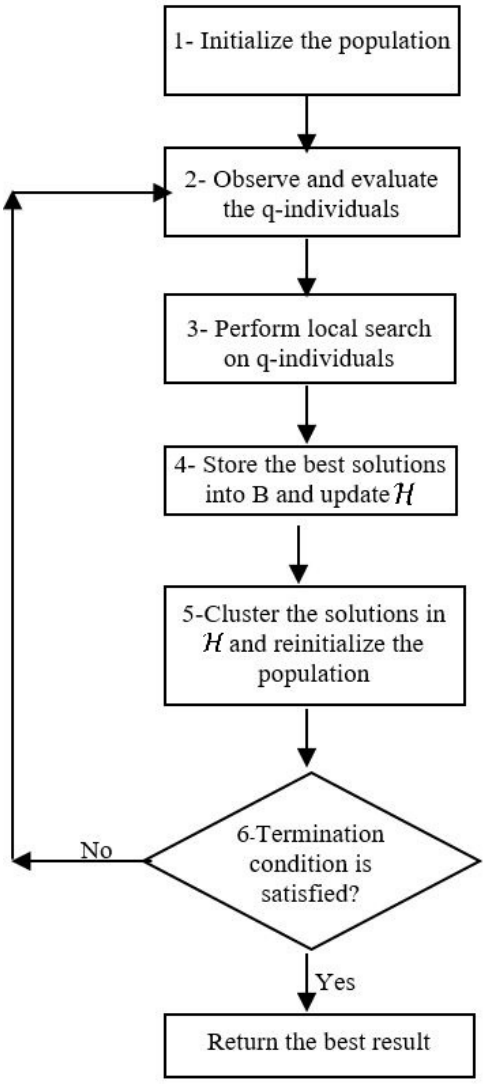

Figure 16: The flowchart of the phase two of the algorithm.

found feasible regions. To do this, when a move is to be taken, the Hamming distance between the new solution to all the solutions in $\mathcal{H}$ is found,

$$\Delta(x, y) = \sum_{c\in C}\sum_{t\in T_s}\sum_{s\in S}\sum_{p\in P}\sum_{d\in D} |x_{ctspd} - y_{ctspd}|, \quad (22)$$

the move is accepted if the local search moves away from the already found feasible regions. If there is no move to satisfy this, a move is taken that does not get closer to the found feasible regions. And if all the moves get closer to the feasible regions, one is chosen randomly. This way, by avoiding the areas in the search space that have already been explored, the local search algorithm tries to find as many feasible regions as possible. Note that in our proposed local search, all the moves decrease the number of hard constraints that are broken; therefore, there will be no loop in the search process.

When all the hard constraints are satisfied, the local search starts optimizing the soft constraints at step 5. Here the algorithm performs random walks and takes steps when a better solution is found. The random walk in this paper is performed by finding all the feasible neighbors to the current solution and moving to one of them randomly. The local search is performed until no better solution is found at the neighborhood.

Steps 13 of the algorithm 2 collects information about the fitness landscape during the search process and tries to capture the structure of the fitness landscape. This information can then be used to guide the search algorithm. To collect the information, the set, $\mathcal{H}$ is devised. Here, the best possible solutions among the binary solutions X^{τ} are added to $\mathcal{H}$, where $\mathcal{H}$ is the history of the solutions found in previous searches until the current iteration, τ. The location of these good solutions and their fitness contains information about the structure of the fitness landscape that can be exploited by optimization algorithms.

Phase 1 of the algorithm only explores the search space and collects information. In phase 2, the collected information

is exploited to guide the search algorithm. This information is used to guide both the local search algorithms and the population of quantum individuals.

In step 23, the convergence status of the population is checked. If the population has converged, it means that the algorithm is trapped in local optima and thus the population is reinitialized based on $\mathcal{H}$.

As the data in Figure 14 demonstrate, good solutions tend to be closer to one another than random solutions. This suggests that good solutions are clustered in the search space. This property has also been observed in other problems Prugel-Bennett and Tayarani-Najaran (2011); Tayarani-N and Prügel-Bennett (2015); Tayarani-N. and Prügel-Bennett (2015); Tayarani-N. and Prugel-Bennett (2014); Tayarani-N. and Prugel-Bennett (2016). Therefore, in step 24 of the algorithm, the best solutions that have been found in the previous searches are clustered to exploit the area around each cluster separately. Here $\mathcal{H}$ contains the best solutions throughout the search process. The k-means clustering algorithm is used in this paper to cluster the solutions in $\mathcal{H}$ to find the center of each cluster, κ_i, for $i = 1 \dots K$ where K is the number of clusters.

In step 24, the solutions in $\mathcal{H}$ are first clustered to generate the sets $\mathcal{C}_k$, for $\kappa = 1 \dots K$, where K is the number of clusters. Then the algorithm divides the population into K groups of q-individuals and assigns a group to each cluster. Each group is then reinitialized around the center of the cluster it represents to search the area around it. The center of each cluster is calculated as the weighted average of the solutions in the cluster. The weight of the solutions is a function of the iteration in which the solutions have been found. This weighted averaging system is used so the more recent solutions have greater weight,

$$\mathcal{B}^{\kappa}_{ctspd} = \frac{1}{\sum_{\forall x \in \mathcal{C}^{\kappa}} e^{-\frac{\tau - \mathcal{T}(x)}{\lambda}}} \sum_{\forall x \in \mathcal{C}^{\kappa}} e^{-\frac{\tau - \mathcal{T}(x)}{\lambda}} x_{ctspd}, \quad (23)$$

$$\forall c \in C, t \in T_s, s \in S, p \in P, d \in D,$$

where $\mathcal{T}(x)$ returns the iteration at which the solution x was found and τ is the current iteration. As per equation 23, the solutions in more recent iterations have more weight and so more influence on $\mathcal{B}_i$. The weight of the solutions decays exponentially with their age. If $\mathcal{B}_i$ is close to 1, it means that in most of better possible solutions, this bit has the value of 1, and it is better to give a value to this q-bit that represents 1 with higher probability. Accordingly, this paper proposes the following method for the reinitialization step,

$$q^{\tau}_{i,ctspd} = \frac{\pi}{4} + \frac{\pi}{3} \times (\mathcal{B}_{ctspd} - \frac{1}{2})$$

$$\forall c \in C, t \in T_s, s \in S, p \in P, d \in D, \quad (24)$$

where $i = 1, 2, ..., m$, m is the number of q-individuals in the population. This formula re-initializes the q-bits between the values of $\pi/4 - \pi/6$ and $\pi/4 + \pi/6$ based on the value of $\mathcal{B}_{ctspd}$.

Table 2
The problem instances used in this paper.

	n_c	n_s	n_t	n_p	n_d	n		n_c	n_s	n_t	n_p	n_d	n
I_1	4	2	4	3	3	32	I_{16}	4	2	4	3	3	25
I_2	4	2	4	3	5	54	I_{17}	4	2	4	3	5	41
I_3	4	2	16	3	5	54	I_{18}	4	2	16	3	5	41
I_4	4	6	8	3	3	32	I_{19}	4	6	8	3	3	25
I_5	4	6	16	5	3	54	I_{20}	4	6	16	5	3	42
I_6	4	6	20	5	5	90	I_{21}	4	6	20	5	5	70
I_7	8	2	12	3	3	64	I_{22}	8	2	12	3	3	50
I_8	8	6	16	3	5	108	I_{23}	8	6	16	3	5	83
I_9	12	2	20	5	5	270	I_{24}	12	2	20	5	5	209
I_{10}	12	4	12	3	5	162	I_{25}	12	4	12	3	5	125
I_{11}	12	6	12	3	3	97	I_{26}	12	6	12	3	3	75
I_{12}	12	6	20	3	3	97	I_{27}	12	6	20	3	3	75
I_{13}	16	2	16	3	3	129	I_{28}	16	2	16	3	3	100
I_{14}	16	2	20	3	5	216	I_{29}	16	2	20	3	5	167
I_{15}	20	4	20	3	5	270	I_{30}	20	4	20	3	5	210

The proposed algorithm collects information from previous searches and builds a database of good solutions. Then uses this information to reinitialize the q-individuals with the values that represent better parts of search space.

5.1. Parameter Study

In order to test the algorithm different problem instances are used. There are two classes of problems used in this paper, the first class is the random problems with different problem sizes. The list of random problem instances is summarized in Table 2. These problems are generated using the method presented in Abramson and Abela (1991). The second class is the benchmark problems. In this paper hdtt hdt, Valouxis Valouxis and Housos (2003), High School Post et al. (2014), South African Primary School Raghavjee and Pillay (2015) and South African High School Raghavjee and Pillay (2015) are used as benchmark problems.

The population size is an important parameter that affects its performance **?**. Thus, one main question is what is the best population size for a certain computational budget? Due to its probabilistic representation, a q-individual in QEA can perform a search through the landscape without having interaction with other q-individuals. This makes QEA different from all the other evolutionary algorithms that use nonprobabilistic representation for solutions. However, having only one q-individual is not a good choice as it sacrifices the exploration ability of the algorithm. To study this we perform an analysis to find the best population. The results on different problems of different sizes are shown in Table 3. The results are averaged over 30 runs. The results are compared for different size of the population for $n = 1, 2, 5, 10, 15, 25, 40, 60, 100, 150$ and 200. The number of the function evaluations for all the population sizes is set to 50000, that is for example for the population size of $n = 25$, the number of iterations is $50000/25 = 2000$.

The data suggest that the best performance is achieved with a population size equal to 25. Here, the population size $n = 1$ does not reach the best performance for any of the

Table 3
Comparison between different size of the population. This is for $m = 100$ and the data are averaged over 30 different runs.

Benchmark	n=1	n=2	n=5	n=10	n=15	n=25	n=40	n=60	n=100	n=150	n=200
hdtt4	69.42	65.75	68.23	67.59	66.16	66.45	66.84	67.15	**65.68**	66.65	68.05
hdtt5	116.72	117.38	116.26	117.21	116.32	115.1	115.39	117.25	**114.76**	116.52	114.93
hdtt6	189.15	179.82	178.62	182.33	**177.72**	180.68	183.79	180.23	182.35	187.81	189.06
hdtt7	278.5	274.43	**273.05**	274.25	275.63	277.07	275.25	277.45	276.36	279.06	278.15
hdtt8	379.05	378.29	377.82	375.8	374.81	**374.31**	374.41	378.88	377.01	377.28	378.74
Valouxis	34.93	34.2	33.59	**30.21**	32.47	32.76	33.1	31.95	33.19	33.5	34.46
HS1	63.54	**60.05**	64.47	60.2	61.76	61.71	60.51	60.2	63.94	62.27	64.76
HS2	64.72	61.27	63.62	63.45	62.24	**60.71**	61.07	62.78	63.24	63.94	64
HS3	15.84	13.73	13.82	13.09	13.29	**12.69**	14.09	14.46	14.72	14.66	15.22
HS4	43.68	43.46	42.06	**41.22**	41.7	42.53	42.05	42.97	44.9	43.49	43.35
HS5	11.4	11.38	11.33	10.63	10.99	**10.09**	10.56	10.64	10.48	10.81	11.75
HS7	98.63	95.02	97.88	96.57	91.86	92.58	97.1	**90.06**	92.51	96.84	98.89
SAPS	3.87	2.63	2.78	3.02	2.44	2.25	2.44	**2.17**	3.14	3.46	4
SAHS	2.57	1.92	1.98	1.33	1.07	1.14	1.39	2.03	**1.02**	1.54	2.57

problems. One explanation could be a large number of local optima in combinatorial optimization algorithms, that make on single q-individual inert (see for example Prugel-Bennett and Tayarani-Najaran (2011)). If the local optima in the landscape are close to one another and the global optimum is nearby, then a single q-individual may have a chance of finding the global optimum by jumping from a local optimum to another until it reaches the global optimum. However, in combinatorial optimization problems in which the local optima are far apart, a single q-individual is not capable of performing the search and there is a need for a population of q-individuals that cooperate and help one another to escape from local optima and reach the global optimum.

When the population is too large $n > 150$, the performance of the algorithm decays. This could be explained by the fact that a larger number of q-individuals results in each q-individual receiving a smaller computational budget, so they have less time to exploit local optima. In other words, the algorithm has too much leaned towards exploration.

The proposed algorithm has two parameters, λ which controls the memory of the algorithm on the best-observed solutions in different iterations, and ρ in step 23 of the algorithm 2 which controls the frequency at which the proposed algorithm is applied to control the exploitation of the collected information and the exploration through the search space. This section analyzes the effect of these two parameters on the performance of the algorithm. A method similar to Tayarani-N. and Akbarzadeh-T. (2014) is used here. The size of the population for all the experiments is set to 25 and the parameters are set to $\rho_1 \dots \rho_5 = (0.8, 0.9, 0.95, 0.99, 1)$, and $\lambda_1 \dots \lambda_5 = (1, 10, 50, 100, 200)$. In order to find the best parameters, we found the performance of the algorithm for all the possible combinations (which is 5×5=25 possibility) and report the values that offer the best performance. For these sets of parameters, we tried to use a domain in which the best value for the parameters lies within the domain. To do so, we use the two extreme values at each end. For example for λ we use 1 at one side and 200 at the other which are the values at the two extreme ends. Here $\rho_5 = 1$ is included which makes the proposed algorithm the same as the simple QEA. This is because if $\rho = 1$ then the population is reinitialized when all the q-individuals are at the true states of $[\alpha \ \ \beta] = [0 \ \ 1]$ or $[\alpha \ \ \beta] = [1 \ \ 0]$ which never occurs. We also tried to use the domain for these parameters in which the graphs representing the performance of the algorithm are as convex as possible. The best parameters for the algorithm are summarized in Table 4. The data in this table show an improvement for the proposed algorithm over the original version of QEA as $\rho = 1$ has not reached the best performance for any of the problem instances. The best parameters for different problems are quite similar, the best value for ρ is around 0.99 and the best value for λ is 100. From now on, we use $\rho = 0.99$ and $\lambda = 100$ for the proposed algorithm.

The proposed algorithm is compared with the original version of QEA, Genetic Algorithm, Differential Evolution, Fast Evolutionary Strategy, Particle Swarm Optimization, Simulated Annealing algorithms, the algorithm proposed in Song et al. (2018), Wang et al. (2007), Dang et al. (2016), Ceschia and Schaerf (2018) and Legrain et al. (2017). In order to perform a fair comparison between the algorithms, for each algorithm, the performance of the algorithms is studied for a set of parameters, and the best parameters for the algorithms are found. The best parameters for different algorithms are summarized in Table 5. These parameters are used in the experiments.

6. Experimental Results

This section performs an experimental study on different algorithms. In all experiments the maximum number of generations is set to 2000, the size of the population is 25 (50,000 function evaluations) and the parameters of the algorithms are set according to Tables 4 and 5. The mode value of the parameters in these tables is used in the experiments. In order to perform a fair comparison, all algorithms have been given an equal number of function evaluations. In population-based algorithms,

Table 4
The best parameters for the proposed algorithm for different problems. The results are averaged over 30 runs.

Random Problems			Random Problems		
	ρ	λ		ρ	λ
I_1	0.99	10	I_{16}	0.95	10
I_2	0.90	100	I_{17}	0.99	100
I_3	0.99	50	I_{18}	0.95	50
I_4	0.99	100	I_{19}	0.90	100
I_5	0.95	100	I_{20}	0.99	50
I_6	0.95	100	I_{21}	0.99	50
I_7	0.90	100	I_{22}	0.99	50
I_8	0.99	50	I_{23}	0.99	50
I_9	0.99	100	I_{24}	0.99	100
I_{10}	0.99	50	I_{25}	0.90	50
I_{11}	0.99	100	I_{26}	0.99	50
I_{12}	0.99	50	I_{27}	0.90	50
I_{13}	0.99	100	I_{28}	0.95	100
I_{14}	0.99	100	I_{29}	0.99	50
I_{15}	0.99	100	I_{30}	0.95	100
Real World Problems			Real World Problems		
	λ	ρ		λ	ρ
hdtt4	0.95	50	HS2	0.90	50
hdtt5	0.90	50	HS3	0.95	50
hdtt6	0.95	100	HS4	0.90	100
hdtt7	0.99	50	HS5	0.90	50
hdtt8	0.99	50	HS7	0.95	50
Valouxis	0.99	50	SAPS	0.99	100
HS1	0.90	100	SAHS	0.95	50

the population size is set to 25 and the number of iterations to 2000. For non-population algorithms, like SA, 50,000 iterations are used so the algorithm receives the same computational budget. Also, the number of clusters in the proposed algorithm is set to $K = 3$ as a gap analysis on the data suggests this is the best representation for the data. Also, some experiments suggest that $K = 3$ performs better than other values. To cluster the data the k-means clustering algorithm is used. Table 6 summarizes the results for different algorithms on the random problem instances and the benchmark problems. The parameters for these algorithms are set to $\rho = 0.99$ and $\lambda = 100$ for the proposed algorithm. The best results in this table are boldfaced. Among all the problems, the proposed algorithm has achieved the best results for 21 problem instances. After the proposed algorithm sits Ceschia and Schaerf (2018) with achieving the best results for 5 problem instances. The next algorithm in the ranking is Dang et al. (2016), which achieves the best performance for 4 problems. Then is GA which achieves the best results for 3 problems.

In order to find the true ranking of the algorithms, the Friedman two-way analysis of variance by ranks test is performed. Table 8 shows the ranking of the algorithms based on the data presented in Table 6. According to Friedman's ranking test, among the algorithms, the proposed algorithm is ranked the best with 4.86. After that is QEA which is ranked 10.52.

The reason why the proposed algorithm and QEA perform better is that these algorithms have particularly been developed for binary-coded combinatorial optimization problems. Therefore, these algorithms inherently are more suitable for the timetabling problems as this problem is, in nature, a combinatorial binary problem. The proposed algorithm performs better than the original version of QEA because in QEA there is no mechanism to help the algorithm explore the search space when the q-individuals are converged and the algorithm is trapped in local optima. In QEA, after some iterations, the q-individuals converge and when this happens, it means that they are trapped in a particular region in the search space and have focused their attention on the close-by local optima. In this situation, the algorithm has little chance of escaping from the local optima and to explore the search space. The proposed algorithm, however, is equipped with a reinitialization mechanism that can detect when the algorithm is trapped in local optima and gives the algorithm another chance to perform a search. Instead of a random reinitialization, the proposed algorithm builds a database of the best solutions it has found during its previous search. This database is analyzed to find the regions in the search space that are more likely to contain better solutions. Also, as the landscape analysis showed in this paper, good solutions are clustered in the landscape. Inspired by this, the proposed algorithm clusters the solutions and creates a sub-population of q-individuals, assigning each sub-population to one cluster. Therefore, the algorithm can balance its computational budget between different clusters of good local optima.

After QEA is GA which is ranked 12.09. GA does not perform as well as QEA, because it is not particularly designed for combinatorial binary problems. However, GA still performs better than other algorithms. This is because this algorithm is well suited for binary problems, and its operators can be more inherently applied to strings of binary solutions. Next among the algorithms are FES, SA, PSO, and DE respectively. Unlike GA and QEA that can inherently represent binary solutions, these algorithms are more suitable for continuous problems and must be tuned for binary solutions. Especially for the case of PSO and DE, in their nature, these algorithms are not as suitable for combinatorial optimization problems and this could explain why they are ranked after QEA and GA.

Among the algorithms proposed in referenced papers, the best performance is achieved by Legrain et al. (2017) which is ranked 11.81. It is interesting that QEA achieves better performance than this algorithm. This indicates the potential of QEA in solving timetabling problems. Also, Song et al. (2018) does not offer good performance compared to QEA and GA. We believe that it is because Song et al. (2018) proposes a local search algorithm for the problem. Local search algorithms are prone to get trapped in local optima, a drawback that is managed in population-based algorithms. The global search that GA and QEA offer, here helps them to perform better. The algorithm proposed by Wang et al. (2007), which is a QEA with PSO update operator, performs worse than the original version of QEA. It is interesting as PSO also does not perform well here. Seems that PSO is not a suitable mechanism for solving the timetabling problem

Table 5
The best parameters for each of the benchmark problems. The number of iterations is 2000 and the data are averaged over 100 different runs.

	PSO			GA		DE		FES		SA	Wang et al. (2007)	
	c_1	c_2	W	M	R	F	O	L	S	η	c_1	c_2
hdtt4	5	0.1	0.1	0.005	1.0	0.1	0.8	1.0	1.1	0.9	1.25	1.75
hdtt5	0.5	0.5	0.1	0.003	1.0	0.1	0.2	1.0	1.1	0.95	1.5	1.25
hdtt6	1.5	1.0	0.7	0.003	1.0	0.5	0.2	1.0	1.1	0.9	1.75	1.25
hdtt7	5	1.0	0.1	0.003	1.0	0.1	0.4	1.0	1.1	0.95	1.5	1.25
hdtt8	5	1.0	0.5	0.003	1.0	0.1	0.4	1.0	1.1	0.9	1.5	1.25
Valouxis	2	0.5	0.7	0.005	1.0	0.5	0.2	1.0	1.1	0.95	1.25	1.25
HS1	2	0.5	0.7	0.003	1.0	0.5	0.2	1.0	1.1	0.95	1.25	1.5
HS2	5	0.5	0.7	0.01	1.0	1.0	0.2	1.0	1.2	0.9	1.25	1.75
HS3	5	1.0	1	0.003	1.0	0.1	0.2	1.0	1.1	0.95	1	1.25
HS4	2	1.5	0.7	0.003	1.0	0.1	0.4	1.0	1.1	0.95	1.25	1.5
HS5	2	0.5	0.7	0.003	1.0	0.5	0.2	1.0	1.1	0.95	1.5	1.75
HS7	2	1.5	0.7	0.005	1.0	0.1	0.2	1.0	1.1	0.95	1.25	1.5
SAPS	5	0.1	0.1	0.003	1.0	0.1	0.2	1.0	1.1	0.9	1.75	1.25
SAHS	5	0.1	0.1	0.003	1.0	0.1	0.4	1.0	1.1	0.95	1.25	1.5

even when it is used in combination with an effective algorithm like QEA.

In order to statistically test the proposed algorithm, Kruskal-Wallis Sheskin (2003) and two-tailed Wilcoxon signed-ranks Wilcoxon (1945) tests are performed between the algorithms. Tables 10 and 11 list the results of the Kruskal-Wallis test between the algorithms where 'SS' is the sum of squares of each source, 'df' is the degree of freedom associated with each source, 'MS' is the mean squares (the ratio SS/df) and 'Chi-square' is the ratio of mean squares. The p-values in these tables show the probability that these samples are taken from populations with the same means. As the data in these tables suggest, the p-values are very small, indicating that the null hypothesis that all the samples are taken from the same mean is rejected significantly.

Figure 17 shows the box plot of the results for different problems. The central mark indicates the median, the top, and bottom edges indicate the 75th and 25th percentiles respectively, the whiskers show the most extreme data, and the outliers are plotted by the '+' symbol. The experiments are performed over 30 runs. This is for the same analysis as performed to collect the data in tables 10 and 11.

Figure 18 shows the cost of the best solution found in each iteration for different problems and algorithms. Here we present three problems as a representative of the general behavior of the algorithms. We observed, to some extent, similar behavior for other problems as well. As seen in this graph, the algorithm proposed in Wang et al. (2007) reaches very quickly a good quality solution but gets trapped in local optima and does not progress anymore. The algorithm uses PSO as an update operator for QEA. The genetic Algorithm on the other hand offers a more steady improvement. It does not converge very quickly, but at the same time does not get trapped in local optima at the early stages of the search process.

In order to statistically contrast the results obtained, the two-tailed Wilcoxon signed-ranks test is used, which is similar to the paired t-test in nonparametric statistical procedures. Wilcoxon test is utilized for determining if there is a significant difference between the two sample means. Table 9 represents the results of the Wilcoxon test for different problems and different algorithms versus the proposed algorithm. In this table, R^+ is the sum of ranks for all the problems in which the proposed algorithm outperforms each of the algorithms and R^- is the sum of ranks for the opposite. The data suggest that the proposed operator can improve the performance of QEA.

7. Conclusion

To develop successful optimization algorithms requires an understanding of the problem structure. This paper performed an analysis of the fitness landscape of the timetabling problem and studied a number of properties of the landscape of the problem. We showed that, similar to some other optimization problems Prugel-Bennett and Tayarani-Najaran (2011); Tayarani-N and Prügel-Bennett (2015), there is a phase transition in the proportion of solvable problem instances in this problem. This phase transition is observed versus the number of classes and the number of time-slots. This paper also studied the landscape ruggedness of the problem. This paper showed that the landscape ruggedness decreases with the number of time-slots.

By studying the time it takes for a local search algorithm to solve hard constraints, we showed how different parameters of the problem can affect the time complexity of the problem. Our observations suggest that the time it takes for a local search to solve the problem grows much faster than exponentially with the number of time-slots as the phase transition is approached.

Quantum Evolutionary Algorithms were first developed for the class of combinatorial optimization problems, and since the timetabling problem is a combinatorial optimization problem, these algorithms should perform well in solving these problems. Nevertheless, not many researchers have

Table 6
The experimental results for different algorithms on different benchmarks. The parameters for each algorithm is set according to Tables 4 and 5 and the results are averaged over 100 runs. The best results are boldfaced.

		QEA	HQEA	GA	PSO	SA	DE	FES	Song et al. (2018)	Wang et al. (2007)	Dang et al. (2016)	Ceschia and Schaerf (2018)	Legrain et al. (2017)
I_1	Mean	48.97	**48.88**	50.18	53.01	52.99	53.02	52.98	50.67	51.14	51.72	52.26	52.74
	STD	0.47	0.6	0.54	0.66	0.43	0.78	0.82	0.82	0.67	0.62	0.74	0.61
I_2	Mean	49.6	**48.71**	49.63	52.1	53.94	53.86	51.12	53.32	51.1	49.01	51.89	49.26
	STD	0.11	0.1	0.09	0.07	0.13	0.09	0.13	0.14	0.14	0.13	0.11	0.1
I_3	Mean	101.29	**98.08**	102.88	104.48	107.07	107.78	104.44	98.45	104.55	104.26	100.69	98.31
	STD	0.64	1.02	0.84	1.09	0.83	0.77	0.84	1.08	0.71	0.71	0.58	0.96
I_4	Mean	58.87	58.19	**57.88**	60.85	63.59	60.61	59.64	60.21	59.8	59.72	60.47	57.91
	STD	0.39	0.45	0.28	0.31	0.43	0.36	0.28	0.42	0.32	0.46	0.28	0.29
I_5	Mean	104.78	**97.37**	107.58	99.35	102.9	99.97	107.56	97.38	104.14	103.37	99.15	97.51
	STD	1.65	0.92	0.86	1.5	1.45	1.46	1.07	1.05	1.35	0.98	0.86	1.44
I_6	Mean	105.85	**97.23**	102.79	105.17	104.97	107.62	97.79	100.23	101.33	100.56	97.34	100.81
	STD	1.25	1.11	1	0.9	1.01	0.93	0.75	1	1.41	0.88	1.17	1.06
I_7	Mean	179.23	170.69	165.2	173.83	172.91	174.64	169.82	179.78	168.71	**162.38**	170.87	175.21
	STD	1.63	1.43	2.46	1.9	1.42	1.49	2.41	2.55	1.34	2.58	2.27	2.46
I_8	Mean	166.5	165.07	163.23	173.86	179.16	175.95	179.33	175.72	166.28	**162.93**	170.06	163.79
	STD	1.62	2.64	2.46	2.22	2.53	2.81	2.44	1.5	2.74	2.21	2.07	2.28
I_9	Mean	153.75	**146.65**	151.23	150.31	161.67	150.16	149.7	157.44	151.29	157.47	152.7	160.69
	STD	1.85	1.47	1.46	1.77	2.78	2.25	1.88	1.76	2.51	1.63	2.88	1.81
I_{10}	Mean	245.74	**243.36**	262.52	258.49	260.98	255.24	267.27	264.17	248.14	266.77	266.29	258.4
	STD	1.54	1.28	1.29	1.37	1.97	1.64	1.52	1.33	1.17	1.38	1.83	1.08
I_{11}	Mean	89.61	89.56	94.16	89.73	95.21	92.09	93.37	89.11	96.71	**87.62**	92.08	94.97
	STD	1.55	1.37	1.09	1.24	0.91	0.83	1.47	1.43	1.54	1.01	1.08	1.51
I_{12}	Mean	92.94	**87.65**	91.16	93.81	91.36	92.5	96.66	89.98	88.03	94.95	88.49	92.03
	STD	0.87	1.41	1.07	1.23	1.45	1.32	1.11	1	1.14	1.36	0.79	0.84
I_{13}	Mean	153.21	149	148.78	160.81	159.91	160.61	157.47	150.43	148.33	147.11	**146.47**	151.44
	STD	1.86	1.64	1.61	1.36	0.96	1.42	1.28	1.59	1.23	1.48	1.77	1.45
I_{14}	Mean	198.83	205.31	**197.13**	211.32	201	209.41	212.21	214.35	206.27	207.63	208.91	200.15
	STD	1.25	1.54	1.42	1.23	1.05	1.63	1.59	1.5	1.95	1.47	1.66	1.07
I_{15}	Mean	340.91	**329.17**	342.69	354.42	333.35	358.13	354.33	331.04	334.42	347.24	350.67	351.49
	STD	3.98	2.96	4.7	3.41	3.42	4.38	3.49	4.83	4.4	4.41	2.64	3.91
I_{16}	Mean	37.63	**37.18**	40.7	39.04	39.08	39.16	40.14	39.59	40.78	39.82	40.55	38.55
	STD	0.45	0.51	0.64	0.55	0.67	0.76	0.42	0.7	0.63	0.7	0.48	0.65
I_{17}	Mean	38.66	37.35	40	40.92	39.79	40.02	40.67	40.58	40.25	39.25	**37.14**	39.69
	STD	0.69	0.5	0.72	0.47	0.62	0.49	0.64	0.46	0.66	0.65	0.62	0.44
I_{18}	Mean	78.61	**76.74**	81.99	83.5	83.67	79.85	83.09	80.09	81.97	81.55	81.73	78.09
	STD	0.9	0.8	1.12	1.18	1.05	1.07	0.94	1.25	0.95	0.89	0.93	1.02
I_{19}	Mean	47.06	**45.07**	49.83	49.07	49.53	48.47	45.76	47.99	49.98	49.24	46.11	47.58
	STD	0.57	0.71	0.54	0.47	0.63	0.4	0.58	0.66	0.59	0.5	0.55	0.36
I_{20}	Mean	77.64	76.26	80.99	82.6	80.19	**75.79**	82.17	77.55	80.08	81.94	81.81	79.56
	STD	0.35	0.4	0.47	0.36	0.29	0.45	0.36	0.45	0.31	0.34	0.43	0.41
I_{21}	Mean	76.49	**76.29**	81.54	77.89	82.1	81.82	82.02	79.95	80.08	80.29	81.07	79.23
	STD	0.97	1.13	0.89	0.74	0.98	0.7	0.9	1.05	1.06	0.94	1.04	1.04
I_{22}	Mean	134.57	130.74	135.05	135.74	134.6	138.42	137.06	136.99	137.06	129.25	**126.45**	133.94
	STD	0.61	0.36	0.51	0.38	0.38	0.35	0.48	0.49	0.45	0.33	0.62	0.64
I_{23}	Mean	128.35	130.35	133.15	134.86	139.94	132.22	139.3	133.1	130.65	**128.11**	135.48	136.48
	STD	1.28	1.94	1.71	2.14	2.41	2.2	1.74	1.35	2.19	2.19	1.77	1.4
I_{24}	Mean	114.77	114.95	119.79	118.19	124.16	123.92	124.38	119.5	114.45	118.83	**113.27**	114.88
	STD	0.61	0.7	0.72	0.44	0.54	0.6	0.68	0.79	0.57	0.67	0.48	0.45
I_{25}	Mean	192.19	**188.94**	189.01	206.33	203.62	198.02	196.02	198.18	193.48	190.03	195.42	192.47
	STD	0.57	0.37	0.51	0.32	0.31	0.56	0.34	0.59	0.36	0.56	0.33	0.48
I_{26}	Mean	74.52	71.57	74.8	74.87	72.56	70.84	73.53	74.7	**69.37**	74.24	74.27	73.86
	STD	0.77	1	0.97	0.87	0.89	0.63	0.88	0.59	0.79	0.73	0.89	0.57
I_{27}	Mean	72.16	**69.25**	71.78	74.5	74.02	74.71	69.85	72.22	72.77	70.16	74.65	70.59
	STD	0.1	0.09	0.09	0.08	0.08	0.07	0.1	0.07	0.07	0.09	0.08	0.09
I_{28}	Mean	118.37	**113.89**	115.39	124.26	113.98	123.95	124.7	116.55	116.51	116.99	115.94	119.26
	STD	0.97	0.73	0.82	1	1.31	1.33	1.23	1.22	1.03	1.1	1.11	1.09
I_{29}	Mean	155.46	152.24	153.42	162.32	157.77	158.84	152.74	159.29	**151.68**	158.49	155.95	154.91
	STD	2	1.19	1.36	1.73	1.78	1.66	1.37	1.58	1.35	1.76	1.59	1.48
I_{30}	Mean	270.62	**254.39**	254.43	262.1	271.19	263.03	256.26	260.83	278.34	279.68	265.84	277.2
	STD	0.47	0.57	0.44	0.44	0.79	0.76	0.8	0.74	0.65	0.53	0.47	0.65

considered solving the timetabling problem with QEA. This paper applied QEA to solving the problem, and in order to improve its performance developed a reinitialization operator.

The advantage of population-based algorithms over local search algorithms is their ability in performing a global search. To benefit from the population of solutions, a successful population-based algorithm should be able to maintain diversity throughout the search process. However, as the search progresses, population-based algorithms lose diversity. One property of the probabilistic representation in QEA is that when the q-individuals are converged, they are trapped around a local optimum, and their chance of escaping from the local optimum decreases. In order to maintain diversity throughout the search process, this paper proposed a reinitialization operator for QEA. In

Table 7
The experimental results for different algorithms on different benchmarks. The parameters for each algorithm is set according to Tables 4 and 5 and the results are averaged over 100 runs. The best results are boldfaced.

		QEA	HQEA	GA	PSO	SA	DE	FES	Song et al. (2018)	Wang et al. (2007)	Dang et al. (2016)	Ceschia and Schaerf (2018)	Legrain et al. (2017)
hdtt4	Mean	**65.32**	66.01	67.88	68.24	66.93	74.67	67.89	72.08	65.66	71.59	71.41	71.32
	STD	0.8	0.78	0.79	0.78	0.83	0.75	0.78	0.74	0.84	0.83	0.78	0.8
hdtt5	Mean	126.97	**116.35**	118.07	123.48	127.87	120.86	119.21	125.7	126.21	125.3	123.19	118.65
	STD	1.65	1.66	1.67	1.69	1.68	1.65	1.65	1.7	1.64	1.67	1.66	1.71
hdtt6	Mean	178.9	181.33	186.97	178.03	188.58	197.1	187.91	188.78	179.52	184.39	**177.57**	185.92
	STD	2.31	2.34	2.33	2.29	2.3	2.29	2.3	2.3	2.27	2.35	2.27	2.34
hdtt7	Mean	**276.78**	281.9	286.48	295.72	290.37	296.69	294.04	282.57	277.5	284.45	288.34	284.79
	STD	3.34	3.31	3.26	3.34	3.29	3.27	3.34	3.26	3.28	3.29	3.26	3.26
hdtt8	Mean	**375.52**	379.54	380.98	398.63	394.19	384.25	381.97	390.21	396.93	390.06	387.3	394.01
	STD	2.23	2.22	2.23	2.24	2.21	2.26	2.26	2.3	2.24	2.23	2.27	2.25
Valouxis	Mean	32.16	31.14	31.91	34.3	**30.74**	36.31	38.82	39.21	37.63	33.09	34.82	37.17
	STD	0.45	0.45	0.45	0.45	0.45	0.48	0.48	0.48	0.47	0.48	0.49	0.48
HS1	Mean	61.24	**60.65**	63.38	66.21	67.67	64.26	64.39	66.29	60.79	66.08	63.71	63.51
	STD	0.66	0.7	0.69	0.66	0.63	0.65	0.68	0.66	0.66	0.66	0.63	0.66
HS2	Mean	61.68	62.33	62.87	64.73	63.54	61.4	61.83	**60.85**	61.05	62.89	63.47	61.83
	STD	0.67	0.79	0.7	0.75	0.76	0.7	0.69	0.69	0.78	0.69	0.72	0.67
HS3	Mean	12.98	13.7	**12.62**	13.87	14.12	13.91	15.94	13.31	13.31	13	14	14.94
	STD	0.27	0.24	0.25	0.25	0.29	0.27	0.26	0.27	0.26	0.25	0.25	0.29
HS4	Mean	41.57	**40.83**	43.13	41.87	43.76	42.28	43.26	41.79	44.27	41.36	42.42	41.93
	STD	0.58	0.57	0.59	0.55	0.55	0.54	0.6	0.59	0.56	0.58	0.6	0.58
HS5	Mean	11.51	**10.41**	11.39	11.56	11.79	11.65	11.29	11.51	11.78	11	11.44	10.76
	STD	0.25	0.26	0.23	0.29	0.22	0.26	0.28	0.28	0.28	0.25	0.24	0.25
HS7	Mean	94.56	90.96	97.43	99.46	92.42	96.18	98.83	92.55	90.27	91.66	92.35	**90.13**
	STD	1.53	1.5	1.51	1.47	1.49	1.5	1.53	1.52	1.53	1.5	1.48	1.48
SAPS	Mean	3.47	3.02	3.03	3	3.49	3.41	**2.29**	3.66	3.39	3.44	3.66	2.44
	STD	0.14	0.13	0.15	0.13	0.14	0.13	0.13	0.15	0.13	0.14	0.15	0.14
SAHS	Mean	1.26	**1.04**	1.14	2.86	1.75	1.97	1.31	1.31	1.38	1.75	1.69	1.2
	STD	0.12	0.12	0.11	0.12	0.12	0.12	0.11	0.11	0.11	0.11	0.11	0.11

Table 8
The Friedman rank of the algorithms based on the data in Table 6 and 7.

	QEA	HQEA	GA	PSO	SA	DE	FES	Song et al. (2018)	Wang et al. (2007)	Dang et al. (2016)	Ceschia and Schaerf (2018)	Legrain et al. (2017)
Rank	10.52	4.857	12.09	18	18.23	17.33	16.33	14.76	13.14	12.80	13.52	11.81

Table 9
The two-tailed Wilcoxon signed ranks test performed between the proposed algorithm and the other algorithms.

Algorithm	R^+	R^-	*p-value*
QEA	805	185	2.97e-04
GA	873	117	1.03e-05
PSO	972	17.5	2.51e-08
SA	968	22	3.39e-08
DE	975	15	2.19e-08
FES	972	17.5	2.51e-08
Song et al. (2018)	969	21	3.17e-08
Wang et al. (2007)	843	147	4.88e-05
Dang et al. (2016)	872	118	1.11e-05
Ceschia and Schaerf (2018)	899	91	2.42e-06
Legrain et al. (2017)	911	79	1.21e-06

the proposed algorithm, the diversity of the population is measured and when the population is converged it is reinitialized so the diversity is preserved. Instead of a random reinitialization, during the search process, the proposed algorithm collects information about the fitness landscape and reinitializes the population with the values to represent better regions in the search space. This way, the proposed algorithm builds a database about the fitness landscape structure and exploits this structure to spend more of its time budget on the area that is more likely to include better solutions.

The proposed algorithm in this paper has two phases. The first phase is the exploration step in which the aim of the algorithm is to collect information about the fitness landscape. In order to collect information from different regions in the search space, a landscape estimating local search is proposed. The proposed algorithm keeps a database of best-observed solutions, and during the search process, uses a tabu mechanism that avoids the regions that have already been explored. This way, the proposed algorithm tries to discover as many feasible regions as it can.

The second phase of the proposed algorithm is the exploitation step, in which the proposed algorithm uses the data it has collected from the fitness landscape to guide the search process. In the proposed algorithm, the best-observed solutions are stored in a database. The analysis showed that there is a correlation between the fitness of a local optimum and its distance to the global optimum. This is a property that was observed in a number of combinatorial optimization problems

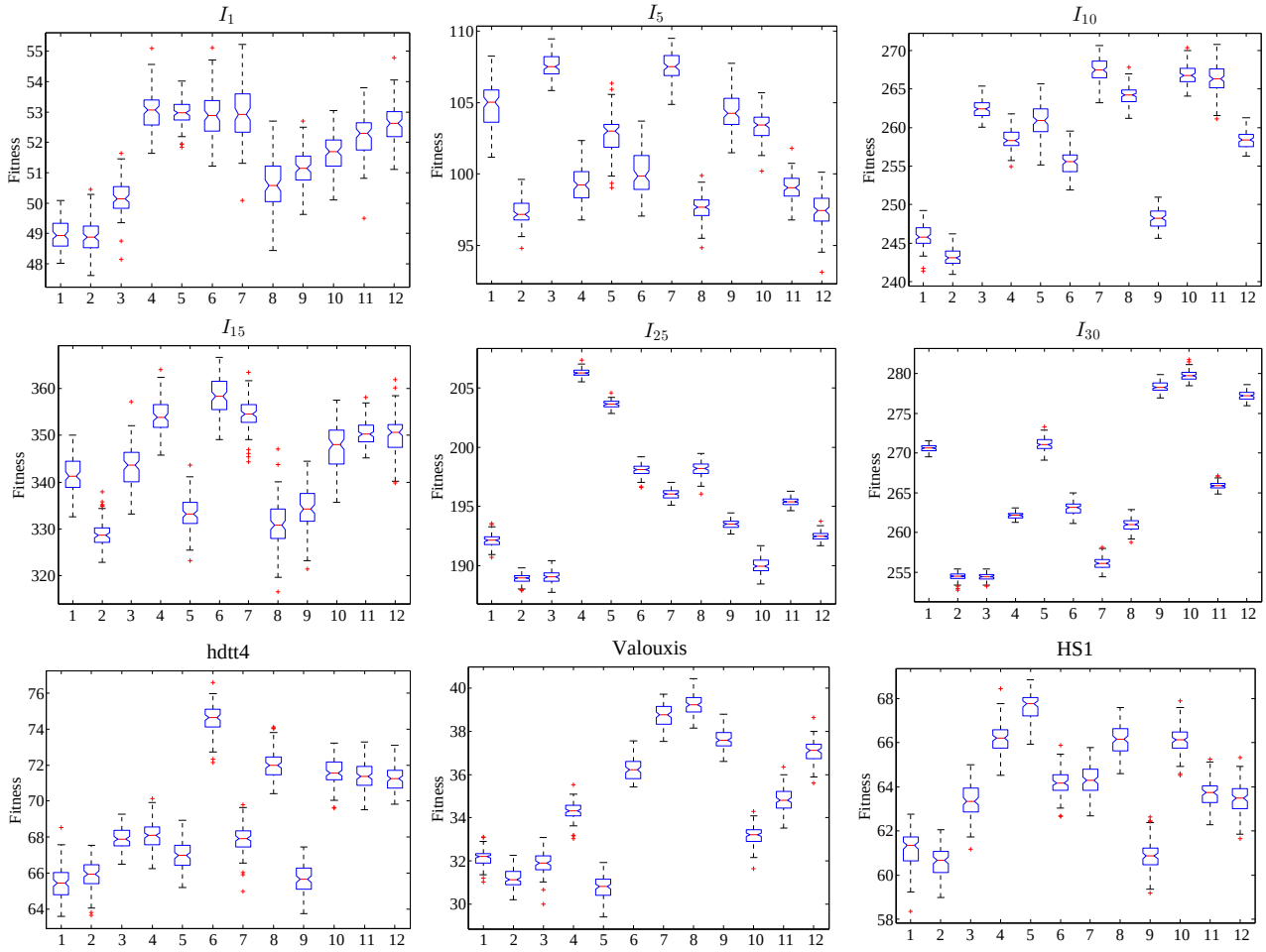

Figure 17: The box plot of the results for different problems. The central mark indicates the median, the top and bottom edges indicate the 75th and 25th percentiles respectively, the whiskers show the most extreme data and the outliers are plotted by '+' symbol. The experiments are performed over 30 runs. The list of algorithms are as follows: 1-QEA, 2-HQEA, 3-GA, 4-PSO, 5-SA, 6-DE, 7-FES, 8-Song et al. (2018), 9-Wang et al. (2007), 10-Dang et al. (2016), 11-Ceschia and Schaerf (2018), 12-Legrain et al. (2017).

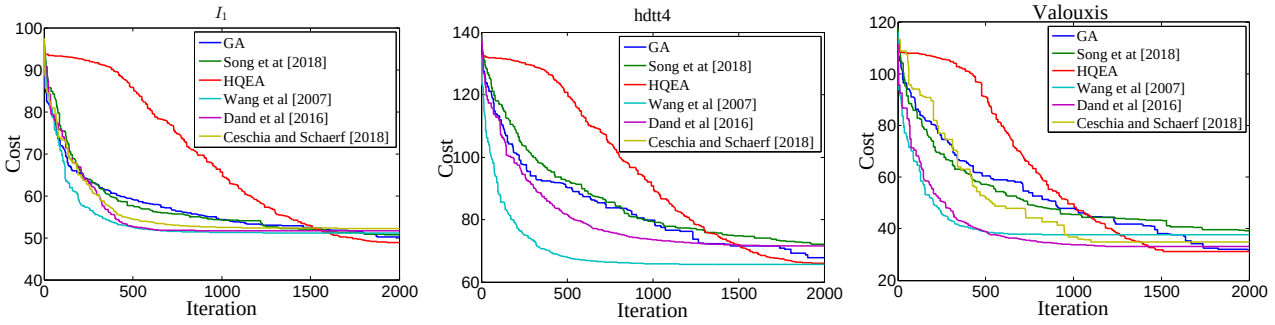

Figure 18: The cost of the best solution found in each iteration for different algorithms and problems.

Prugel-Bennett and Tayarani-Najaran (2011); Tayarani-N and Prügel-Bennett (2015); Tayarani-N. and Prügel-Bennett (2015); Tayarani-N. and Prugel-Bennett (2014); Tayarani-N. and Prugel-Bennett (2016). This suggests that good solutions in the search space are clustered. Therefore, the proposed algorithm clusters the best solutions that are found in previous steps of the algorithm. This way, the search space is divided into sub-regions. Then the population of q-individuals is divided into subpopulations, and each subpopulation is dedicated to a sub-region in the search space.

We tested our proposed method on a number of benchmark and random problems and showed that it improves the performance of QEA in solving the timetabling problems.

In this paper, we showed how the advantages of QEA

can be exploited in solving the timetabling problems. Other optimization algorithms also have their own strengths. One line of future work is to discover the possibility of hybridizing QEA with other optimization algorithms to benefit from the advantages of all these algorithms. Hybrid optimization algorithms are an interesting field of study which have not been explored for solving timetabling using QEA yet. Another future work is to study cooperative evolution in solving timetabling via QEA. In cooperative evolution, the problems are decomposed into smaller pieces and each piece is solved cooperatively with a number of evolutionary processes. In this paper, we initialize the population randomly in a way that the probability of zeros and ones in the string of solution is equal. It is clear that some prior knowledge about the problem can provide insight on how to initialize populations in a way that the algorithm has a better chance of finding better solutions. For example to discover the best probability of setting values to zeros or ones as the solutions may tend to have more zeros or ones. As future work, studying the optimized way of initialization can be considered. In this paper, we showed how studying the fitness landscape analysis can help in solving the timetabling problems via QEA. Another line of research for the future is to explore the ways in which fitness landscape analysis can be used in solving other optimization problems via other optimization algorithms.

8. Appendix

The results for Kruskal-Wallis are summarized in Tables 10 and 11.

References

, . Or-library. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/tableinfo.html. Accessed: 2010-09-30.

Abramson, D., Abela, J., 1991. A parallel genetic algorithm for solving the school timetabling problem. Division of Information Technology, CSIRO.

Akkan, C., Gulcu, A., 2018. A bi-criteria hybrid genetic algorithm with robustness objective for the course timetabling problem. Computers and Operations Research 90, 22 – 32.

Al-Yakoob, S.M., Sherali, H.D., 2015. Mathematical models and algorithms for a high school timetabling problem. Computers and Operations Research 61, 56 – 68.

Alander, J., Zinchenko, L., Sorokin, S., 2002. Analysis of fitness landscape properties for evolutionary antenna design, in: Artificial Intelligence Systems, 2002. (ICAIS 2002). 2002 IEEE International Conference on, pp. 363 – 368. doi:10.1109/ICAIS.2002.1048128.

Alander, J.T., 1999. Practical Handbook of Genetic Algorithms: Complex Coding Systems, Chapter 13, Population size, building blocks, fitness landscape and genetic algorithm search efficiency in combinatorial optimization: An empirical study. volume 3. CRC press.

Angel, E., Zissimopoulos, V., 1998. Autocorrelation coefficient for the graph bipartitioning problem. Theoretical Computer Science 191, 229–243.

Angel, E., Zissimopoulos, V., 2001. On the landscape ruggedness of the quadratic assignment problem. Theoretical Computer Science 263, 159 – 172. doi:10.1016/S0304-3975(00)00239-5.

Babaei, H., Karimpour, J., Hadidi, A., 2015. A survey of approaches for university course timetabling problem. Computers and Industrial Engineering 86, 43 – 59. Applications of Computational Intelligence and Fuzzy Logic to Manufacturing and Service Systems.

Barbosa, V.C., Ferreira, R.G., 2004. On the phase transitions of graph coloring and independent sets. Physica A: Statistical Mechanics and its Applications 343, 401 – 423. doi:10.1016/j.physa.2004.05.055.

Boese, K.D., 1995. Cost versus distance in the travelling salesman problem. Technical Report. UCLA computer science department, Los Angeles.

Bouziri, H., Mellouli, K., Talbi, E.G., 2009. Fitness Landscape Analysis for Optimum Multiuser Detection Problem. Journal of Combinatorial Optimization 21, 306–329.

Bouziri, H., Mellouli, K., Talbi, E.G., 2011. The k-coloring fitness landscape. Journal of Combinatorial Optimization 21, 306–329.

Burke, E.K., Causmaecker, P.D., Berghe, G.V., Landeghem, H.V., 2004. The state of the art of nurse rostering. Journal of Scheduling 7, 441–499.

Burke, E.K., Petrovic, S., 2002. Recent research directions in automated timetabling. European Journal of Operational Research 140, 266–280.

Caprara, A., Monaci, M., Toth, P., Guida, P.L., 2006. A lagrangian heuristic algorithm for a real-world train timetabling problem. Journal of Discrete Applied Mathematics 154, 738–753.

Ceschia, S., Schaerf, A., 2018. Solving the inrc-ii nurse rostering problem by simulated annealing based on large neighborhoods, PATAT.

Cheang, B., Li, H., Lim, A., Rodrigues, B., 2003. Nurse rostering problems: A bibliographic survey. European Journal of Operational Research 151, 447–460.

Cheeseman, P., Kanefsky, R., Taylor, W.M., 1991. Where the really hard problems are, in: Proceedings of the 12th international joint conference on Artificial intelligence - Volume 1, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 331–337.

Chicano, F., Luque, G., Alba, E., 2010. Elementary landscape decomposition of the quadratic assignment problem, in: Proceedings of the 12th annual conference on Genetic and evolutionary computation, ACM, New York, NY, USA. pp. 1425–1432.

Collard, P., Vérel, S., Clergue, M., 2007. Local search heuristics: Fitness cloud versus fitness landscape. CoRR abs/0709.4010.

Coppersmith, D., Gamarnik, D., Hajiaghayi, T., Sorkin, G., 2004. Random max sat, random max cut, and their phase transitions. Random structures and algorithms 24, 502–545.

Culberson, J., Gent, I., 2001. Frozen development in graph coloring. Theor. Comput. Sci. 265, 227–264.

Czogalla, J., 2008. Fitness landscape analysis for the continuous flow-shop scheduling problem, in: Proceedings of 3rd European Workshop, Evo, Naples.

Czogalla, J., Fink, A., 2009. Fitness Landscape Analysis for the Resource Constrained Project Scheduling Problem. Lecture Notes in Computer Science **5851**, Springer, Berlin.

Czogalla, J., Fink, A., 2011. Fitness landscape analysis for the no-wait flow-shop scheduling problem. Journal of Heuristics , 1–27.

Dang, N.T.T., Ceschia, S., Schaerf, A., De Causmaecker, P., Haspeslagh, S., 2016. Solving the multi-stage nurse rostering problem, in: Proceedings of the 11th international conference of the practice and theory of automated timetabling, pp. 473–475.

Daolio, F., Tomassini, M., Vérel, S., Ochoa, G., 2012. Communities of minima in local optima networks of combinatorial spaces. CoRR 4445.

Daolio, F., Verel, S., Ochoa, G., Tomassini, M., 2010. Local optima networks of the quadratic assignment problem, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, IEEE. pp. 1–8.

Detienne, B., Peridy, L., Pinson, E., Rivreau, D., 2009. Cut generation for an employee timetabling problem. European Journal of Operational Research 197, 1178–1184.

Donati, A.V., Darley, V., Ramachandran, B., 2008. An ant-bidding algorithm for multistage flowshop scheduling problem: Optimization and phase transitions, in: Advances in Metaheuristics for Hard Optimization. Springer Berlin Heidelberg. Natural Computing Series, pp. 111–136.

Easton, K., Nemhauser, G., Trick, M., 2004. Sports scheduling, in: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press. chapter 52.

Eley, M., 2006. Ant algorithms for the exam timetabling problem, in:

Table 10
ANOVA and Kruskal-Wallis test on the data for different problems.

	Source	SS	df	MS	F	Prob > F	Source	SS	df	MS	Chi-sq	Prob>Chi-sq
I_1	Columns	2.616e+03	11	2.378e+02	5.328e+02	0	Columns	1.149e+08	11	1.045e+07	957	3.236e-198
	Error	5.303e+02	1188	4.464e-01			Error	2.905e+07	1188	2.445e+04		
	Total	3.146e+03	1199				Total	1.440e+08	1199			
I_2	Columns	4.015e+03	11	3.650e+02	2.900e+04	0	Columns	1.412e+08	11	1.284e+07	1176	2.715e-245
	Error	1.495e+01	1188	1.258e-02			Error	2.791e+06	1188	2.350e+03		
	Total	4.030e+03	1199				Total	1.440e+08	1199			
I_3	Columns	1.218e+04	11	1.107e+03	1.520e+03	0	Columns	1.323e+08	11	1.203e+07	1102	2.297e-229
	Error	8.650e+02	1188	7.281e-01			Error	1.167e+07	1188	9.824e+03		
	Total	1.304e+04	1199				Total	1.440e+08	1199			
I_4	Columns	2.813e+03	11	2.557e+02	2.011e+03	0	Columns	1.308e+08	11	1.189e+07	1089	1.100e-226
	Error	1.511e+02	1188	1.272e-01			Error	1.317e+07	1188	1.108e+04		
	Total	2.964e+03	1199				Total	1.440e+08	1199			
I_5	Columns	1.597e+04	11	1.452e+03	1.041e+03	0	Columns	1.282e+08	11	1.165e+07	1067	5.742e-222
	Error	1.656e+03	1188	1.394e+00			Error	1.580e+07	1188	1.330e+04		
	Total	1.763e+04	1199				Total	1.440e+08	1199			
I_6	Columns	1.348e+04	11	1.226e+03	1.184e+03	0	Columns	1.311e+08	11	1.192e+07	1091	4.065e-227
	Error	1.230e+03	1188	1.035e+00			Error	1.292e+07	1188	1.088e+04		
	Total	1.471e+04	1199				Total	1.440e+08	1199			
I_7	Columns	2.914e+04	11	2.649e+03	6.368e+02	0	Columns	1.232e+08	11	1.120e+07	1026	5.892e-213
	Error	4.943e+03	1188	4.160e+00			Error	2.082e+07	1188	1.753e+04		
	Total	3.408e+04	1199				Total	1.440e+08	1199			
I_8	Columns	4.472e+04	11	4.066e+03	8.023e+02	0	Columns	1.234e+08	11	1.122e+07	1028	2.043e-213
	Error	6.020e+03	1188	5.067e+00			Error	2.057e+07	1188	1.731e+04		
	Total	5.074e+04	1199				Total	1.440e+08	1199			
I_9	Columns	2.344e+04	11	2.131e+03	4.968e+02	0	Columns	1.149e+08	11	1.044e+07	957	4.250e-198
	Error	5.096e+03	1188	4.290e+00			Error	2.912e+07	1188	2.451e+04		
	Total	2.854e+04	1199				Total	1.440e+08	1199			
I_{10}	Columns	7.660e+04	11	6.964e+03	3.333e+03	0	Columns	1.360e+08	11	1.237e+07	1133	5.248e-236
	Error	2.482e+03	1188	2.089e+00			Error	7.968e+06	1188	6.707e+03		
	Total	7.908e+04	1199				Total	1.440e+08	1199			
I_{11}	Columns	9.256e+03	11	8.414e+02	5.043e+02	0	Columns	1.208e+08	11	1.098e+07	1005	1.309e-208
	Error	1.982e+03	1188	1.668e+00			Error	2.325e+07	1188	1.957e+04		
	Total	1.124e+04	1199				Total	1.440e+08	1199			
I_{12}	Columns	8.536e+03	11	7.760e+02	5.395e+02	0	Columns	1.199e+08	11	1.090e+07	998	4.289e-207
	Error	1.709e+03	1188	1.438e+00			Error	2.409e+07	1188	2.028e+04		
	Total	1.024e+04	1199				Total	1.440e+08	1199			
I_{13}	Columns	3.350e+04	11	3.045e+03	1.369e+03	0	Columns	1.257e+08	11	1.143e+07	1047	1.684e-217
	Error	2.643e+03	1188	2.225e+00			Error	1.829e+07	1188	1.539e+04		
	Total	3.614e+04	1199				Total	1.440e+08	1199			
I_{14}	Columns	3.532e+04	11	3.211e+03	1.384e+03	0	Columns	1.322e+08	11	1.202e+07	1101	3.921e-229
	Error	2.757e+03	1188	2.321e+00			Error	1.180e+07	1188	9.933e+03		
	Total	3.808e+04	1199				Total	1.440e+08	1199			
I_{15}	Columns	1.113e+05	11	1.012e+04	6.576e+02	0	Columns	1.236e+08	11	1.124e+07	1029	1.077e-213
	Error	1.828e+04	1188	1.539e+01			Error	2.041e+07	1188	1.718e+04		
	Total	1.296e+05	1199				Total	1.440e+08	1199			
I_{16}	Columns	1.486e+03	11	1.351e+02	3.770e+02	0	Columns	1.108e+08	11	1.007e+07	922	9.357e-191
	Error	4.258e+02	1188	3.584e-01			Error	3.322e+07	1188	2.796e+04		
	Total	1.912e+03	1199				Total	1.440e+08	1199			
I_{17}	Columns	1.620e+03	11	1.472e+02	4.411e+02	0	Columns	1.032e+08	11	9.378e+06	859	4.243e-177
	Error	3.965e+02	1188	3.338e-01			Error	4.085e+07	1188	3.438e+04		
	Total	2.016e+03	1199				Total	1.440e+08	1199			
I_{18}	Columns	5.342e+03	11	4.856e+02	4.854e+02	0	Columns	1.152e+08	11	1.047e+07	959	1.089e-198
	Error	1.189e+03	1188	1.001e+00			Error	2.879e+07	1188	2.423e+04		
	Total	6.531e+03	1199				Total	1.440e+08	1199			
I_{19}	Columns	3.155e+03	11	2.868e+02	9.554e+02	0	Columns	1.278e+08	11	1.162e+07	1064	2.654e-221
	Error	3.566e+02	1188	3.002e-01			Error	1.617e+07	1188	1.361e+04		
	Total	3.512e+03	1199				Total	1.440e+08	1199			
I_{20}	Columns	6.185e+03	11	5.623e+02	3.839e+03	0	Columns	1.367e+08	11	1.242e+07	1138	3.808e-237
	Error	1.740e+02	1188	1.464e-01			Error	7.333e+06	1188	6.172e+03		
	Total	6.359e+03	1199				Total	1.440e+08	1199			
I_{21}	Columns	4.677e+03	11	4.252e+02	4.679e+02	0	Columns	1.114e+08	11	1.013e+07	928	7.432e-192
	Error	1.080e+03	1188	9.088e-01			Error	3.260e+07	1188	2.744e+04		
	Total	5.757e+03	1199				Total	1.440e+08	1199			
I_{22}	Columns	1.410e+04	11	1.282e+03	5.620e+03	0	Columns	1.366e+08	11	1.242e+07	1138	4.172e-237
	Error	2.710e+02	1188	2.281e-01			Error	7.355e+06	1188	6.191e+03		
	Total	1.437e+04	1199				Total	1.440e+08	1199			

Proceedings of the 6th international conference on Practice and theory of automated timetabling VI, pp. 364–382.

Fonlupt, C., Robilliard, D., Preux, P., 1997. Fitness landscape and the behavior of heuristics, in: in Evolution Artificielle 97 (EA'97.

Forrest, S., Mitchell, M., 1993. What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. Machine Learning 13, 285–319.

Goh, S.L., Kendall, G., Sabar, N.R., 2017. Improved local search approaches to solve the post enrolment course timetabling problem. European Journal of Operational Research 261, 17 – 29.

Grover, L.K., 1992. Local search and the local structure of NP-complete problems. Operations Research Letters 12, 235–243.

Hamiez, J.P., Hao, J.K., 2001. An analysis of solution properties of the graph coloring problem, in: 4th metaheuristics international conference, Porto, Portugal.

Han, K.H., Kim, J.H., 2002. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Transactions on Evolutionary Computation 6, 580 – 593.

Table 11
ANOVA and Kruskal-Wallis test on the data for different problems.

	Source	SS	df	MS	F	Prob>F	Source	SS	df	MS	Chi-sq	Prob>Chi-sq
	Columns	1.569e+04	11	1.427e+03	3.949e+02	0	Columns	1.145e+08	11	1.041e+07	954	1.755e-197
I_{23}	Error	4.292e+03	1188	3.612e+00			Error	2.946e+07	1188	2.480e+04		
	Total	1.998e+04	1199				Total	1.440e+08	1199			
	Columns	1.810e+04	11	1.646e+03	4.370e+03	0	Columns	1.337e+08	11	1.216e+07	1114	6.547e-232
I_{24}	Error	4.475e+02	1188	3.766e-01			Error	1.025e+07	1188	8.629e+03		
	Total	1.855e+04	1199				Total	1.440e+08	1199			
	Columns	3.350e+04	11	3.045e+03	1.787e+04	0	Columns	1.410e+08	11	1.282e+07	1174	7.372e-245
I_{25}	Error	2.025e+02	1188	1.704e-01			Error	3.033e+06	1188	2.553e+03		
	Total	3.370e+04	1199				Total	1.440e+08	1199			
	Columns	3.617e+03	11	3.288e+02	5.104e+02	0	Columns	1.046e+08	11	9.512e+06	871	9.689e-180
I_{26}	Error	7.654e+02	1188	6.443e-01			Error	3.937e+07	1188	3.314e+04		
	Total	4.383e+03	1199				Total	1.440e+08	1199			
	Columns	4.245e+03	11	3.859e+02	5.879e+04	0	Columns	1.419e+08	11	1.290e+07	1181	1.791e-246
I_{27}	Error	7.798e+00	1188	6.564e-03			Error	2.133e+06	1188	1.796e+03		
	Total	4.253e+03	1199				Total	1.440e+08	1199			
	Columns	1.666e+04	11	1.515e+03	1.178e+03	0	Columns	1.235e+08	11	1.123e+07	1028	1.607e-213
I_{28}	Error	1.527e+03	1188	1.286e+00			Error	2.051e+07	1188	1.726e+04		
	Total	1.819e+04	1199				Total	1.440e+08	1199			
	Columns	1.174e+04	11	1.067e+03	4.196e+02	0	Columns	1.154e+08	11	1.049e+07	961	4.616e-199
I_{29}	Error	3.022e+03	1188	2.544e+00			Error	2.858e+07	1188	2.406e+04		
	Total	1.476e+04	1199				Total	1.440e+08	1199			
	Columns	9.306e+04	11	8.460e+03	2.171e+04	0	Columns	1.411e+08	11	1.283e+07	1175	3.588e-245
I_{30}	Error	4.629e+02	1188	3.896e-01			Error	2.859e+06	1188	2.406e+03		
	Total	9.352e+04	1199				Total	1.440e+08	1199			
	Columns	1.005e+04	11	9.141e+02	1.430e+03	0	Columns	1.294e+08	11	1.176e+07	1077	4.258e-224
hdtt4	Error	7.596e+02	1188	6.394e-01			Error	1.461e+07	1188	1.230e+04		
	Total	1.081e+04	1199				Total	1.440e+08	1199			
	Columns	1.734e+04	11	1.577e+03	5.533e+02	0	Columns	1.202e+08	11	1.093e+07	1001	1.186e-207
hdtt5	Error	3.385e+03	1188	2.849e+00			Error	2.378e+07	1188	2.002e+04		
	Total	2.073e+04	1199				Total	1.440e+08	1199			
	Columns	3.689e+04	11	3.354e+03	6.229e+02	0	Columns	1.187e+08	11	1.079e+07	988	6.110e-205
hdtt6	Error	6.396e+03	1188	5.384e+00			Error	2.530e+07	1188	2.129e+04		
	Total	4.329e+04	1199				Total	1.440e+08	1199			
	Columns	4.513e+04	11	4.103e+03	3.643e+02	0	Columns	1.123e+08	11	1.021e+07	935	1.479e-193
hdtt7	Error	1.338e+04	1188	1.126e+01			Error	3.165e+07	1188	2.664e+04		
	Total	5.851e+04	1199				Total	1.440e+08	1199			
	Columns	5.975e+04	11	5.432e+03	1.005e+03	0	Columns	1.308e+08	11	1.189e+07	1089	1.470e-226
hdtt8	Error	6.421e+03	1188	5.405e+00			Error	1.324e+07	1188	1.114e+04		
	Total	6.617e+04	1199				Total	1.440e+08	1199			
	Columns	1.003e+04	11	9.118e+02	3.998e+03	0	Columns	1.389e+08	11	1.263e+07	1157	3.314e-241
Valouxis	Error	2.709e+02	1188	2.281e-01			Error	5.069e+06	1188	4.267e+03		
	Total	1.030e+04	1199				Total	1.440e+08	1199			
	Columns	5.887e+03	11	5.352e+02	1.216e+03	0	Columns	1.304e+08	11	1.185e+07	1086	6.720e-226
HS1	Error	5.227e+02	1188	4.400e-01			Error	1.360e+07	1188	1.145e+04		
	Total	6.409e+03	1199				Total	1.440e+08	1199			
	Columns	1.507e+03	11	1.370e+02	2.593e+02	1.96e-306	Columns	9.948e+07	11	9.044e+06	828	1.580e-170
HS2	Error	6.277e+02	1188	5.284e-01			Error	4.452e+07	1188	3.748e+04		
	Total	2.135e+03	1199				Total	1.440e+08	1199			
	Columns	9.152e+02	11	8.320e+01	1.228e+03	0	Columns	1.259e+08	11	1.145e+07	1048	7.465e-218
HS3	Error	8.046e+01	1188	6.773e-02			Error	1.809e+07	1188	1.523e+04		
	Total	9.956e+02	1199				Total	1.440e+08	1199			
	Columns	1.217e+03	11	1.106e+02	3.470e+02	0	Columns	1.083e+08	11	9.849e+06	902	2.258e-186
HS4	Error	3.788e+02	1188	3.189e-01			Error	3.566e+07	1188	3.002e+04		
	Total	1.596e+03	1199				Total	1.440e+08	1199			
	Columns	1.930e+02	11	1.754e+01	2.547e+02	3.53e-303	Columns	8.946e+07	11	8.133e+06	745	1.268e-152
HS5	Error	8.183e+01	1188	6.888e-02			Error	5.454e+07	1188	4.591e+04		
	Total	2.748e+02	1199				Total	1.440e+08	1199			
	Columns	1.278e+04	11	1.162e+03	5.282e+02	0	Columns	1.158e+08	11	1.053e+07	964	1.012e-199
HS7	Error	2.613e+03	1188	2.200e+00			Error	2.821e+07	1188	2.375e+04		
	Total	1.540e+04	1199				Total	1.440e+08	1199			
	Columns	2.253e+02	11	2.048e+01	1.014e+03	0	Columns	1.208e+08	11	1.099e+07	1006	8.831e-209
SAPS	Error	2.400e+01	1188	2.020e-02			Error	2.315e+07	1188	1.949e+04		
	Total	2.493e+02	1199				Total	1.440e+08	1199			
	Columns	2.764e+02	11	2.512e+01	1.845e+03	0	Columns	1.255e+08	11	1.141e+07	1045	4.847e-217
SAHS	Error	1.618e+01	1188	1.362e-02			Error	1.854e+07	1188	1.561e+04		
	Total	2.925e+02	1199				Total	1.440e+08	1199			

Hartmann, A.K., Weight, M., 2005. Phase Transitions in Combinatorial Optimization Ptoblems. Wiley-Vch.

Herroelen, W., De Reyck, B., 1998. Phase transitions in project scheduling. Open Access publications from Katholieke Universiteit Leuven. Katholieke Universiteit Leuven.

Hertz, A., Jaumard, B., de Aragão, M.P., 1994a. Local optima topology for the k-coloring problem. Discrete Appl. Math. 49, 257–280.

Hertz, A., Jaumard, B., de Aragao, M.P., 1994b. Local optima topology for k-coloring problem. Discrete Applied Mathematics 49, 257–280.

Horn, J., Goldberg, D.E., 1995. Genetic algorithm difficulty and the modality of fitness landscapes. Foundations of Genetic Algorithms 3.

Hornby, G.S., 1996. The recombination operator, its correlation to the fitness landscape and search performance, in: Master of Science Thesis, University of Alberta.

Hoshino, T., Mitsumoto, D., Nagano, T., 1998. Fractal fitness landscape and loss of robustness in evolutionary robot navigation. Auton. Robots 5, 199–213.

Huang, D., Shen, Z., Miao, C., Leung, C., 2009. Fitness landscape analysis

for resource allocation in multiuser ofdm based cognitive radio systems. SIGMOBILE Mob. Comput. Commun. Rev. 13, 26–36.

Huanga, D., Shenb, Z., Miaoa, C., Leungc, C., 2009. Fitness Landscape Analysis for Resource Allocation in Multiuser OFDM Based Cognitive Radio Systems. Mobile Computing and Communications Review 13, 26–36.

Huberman, B.A., T, H., 1987. Phase transitions in artificial intelligence systems. Artificial Intelligence 33, 155–171. doi:10.1016/0004-3702(87)90033-6.

Jones, T., Forrest, S., 1995. Fitness distance correlation as a measure of problem difficulty for genetic algorithms, in: Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 184–192.

Kwan, R., 2004. Bus and train driver scheduling, in: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press. chapter 51.

Lefticaru, R., Ipate, F., 2008. A comparative landscape analysis of fitness functions for search-based testing, in: IEEE 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, USA.

Legrain, A., Omer, J., Rosat, S., 2017. A rotation-based branch-and-price approach for the nurse scheduling problem. Mathematical Programming Computation , 1–34.

Lehn, R., Kuntz, P., 2001. A contribution to the study of the fitness landscape for a graph drawing problem, in: Boers, E. (Ed.), Applications of Evolutionary Computing. Springer Berlin / Heidelberg. volume 2037 of *Lecture Notes in Computer Science*, pp. 172–181.

Leite, N., Melicio, F., Rosa, A.C., 2019. A fast simulated annealing algorithm for the examination timetabling problem. Expert Systems with Applications 122, 137 – 151.

Lewis, R., Thompson, J., 2015. Analysing the effects of solution space connectivity with an effective metaheuristic for the course timetabling problem. European Journal of Operational Research 240, 637 – 648.

Lu, G., Li, J., Yao, X., 2011. Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms, in: Proceedings of the 11th European conference on Evolutionary computation in combinatorial optimization, Springer-Verlag, Berlin, Heidelberg. pp. 108–117.

Manderick, B., de Weger, M., Spiessens, P., 1991. The genetic algorithm and the structure of the fitness landscape, in: Proceedings of 4th International Conference on Genetic Algorithms, pp. 143–150.

Martin, W., Barker, A., Cohoon, J., 1999. Problem perturbation: implications on the fitness landscape, in: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on.

Mathias, K., Whitley, D., 1992. Genetic operators, the fitness landscape and the traveling salesman problem, in: Parallel Problem Solving from Nature, Elsevier Science Publishers. pp. 219–228.

McCarthy, I.P., 2008. Manufacturing strategy: understanding the fitness landscape. International Journal of Operations and Production Management 24, 124–150.

Merz, P., 2004. Advanced fitness landscape analysis and the performance of memetic algorithms. Evol. Comput. 12, 303–325.

Merz, P., B.Freisleben, 2000. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Transactions on Evolutionary Computation 4, 337 – 352. doi:10.1109/4235.887234.

Merz, P., Freisleben, B., 1998. Memetic algorithms and the fitness landscape of the graph bi-partitioning problem, in: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, Springer-Verlag. pp. 765–774.

Mézard, M., Parisi, G., Virasoro, M.A., 1987. Spin-Glass Theory and Beyond. volume 9 of *Lecture Notes in Physics*. World Scientific, Singapore.

Mizuno, K., Hayashimoto, A., Nishihara, S., 2000. Analysis of phase transitions in graph-coloring problems based on constraint structures, in: PRICAI 2000 Topics in Artificial Intelligence. Springer Berlin / Heidelberg. volume 1886 of *Lecture Notes in Computer Science*, pp. 792–792.

Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., Troyansky, L., 1999. Determining computational complexity from characteristic 'phase transisition'. Nature 400, 133–137.

Moscato, P., 1989. On evolution, search, optimisation, genetic algorithms and martial arts: Toward Memetic algorithms. Technical Report. California Institute of Technology, Pasadena.

Moscato, P., Norman, M.G., 1992. A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimisation on message-passing systems, in: Proceedings of the International Conference on Parallel Computing and Transputer Applications, pp. 177–186.

Newth, D., Brede, M., 2006. Fitness Landscape Analysis and optimisation of Coupled Oscillators. Journal of Complex Systems 16, 317–331.

Petrovic, S., Burke, E.K., 2004. University timetabling, in: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press. chapter 45.

Pillay, N., Özcan, E., 2019. Automated generation of constructive ordering heuristics for educational timetabling. Annals of Operations Research 275, 181–208.

Post, G., Kingston, J.H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H., Schaerf, A., 2014. Xhstt: an xml archive for high school timetabling problems in different countries. Annals of Operations Research 218, 295–301.

Pošík, P., Franc, V., 2007. Estimation of fitness landscape contours in eas, in: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM, New York, NY, USA. pp. 562–569.

Prugel-Bennett, A., Tayarani-Najaran, M.H., 2011. Maximum satisfiability: Anatomy of the fitness landscape for a hard combinatorial optimization problem. IEEE transactions on evolutionary computation 16, 319–338.

Qasem, M., Prügel-Bennett, A., . Complexity of max-sat using stochastic algorithms, in: Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008, ACM. pp. 615–616.

Qasem, M., Prügel-Bennett, A., 2010. Learning the large-scale structure of the max-sat landscape using populations. IEEE Transactions on Evolutionary Computation 14, 518–529.

Raghavjee, R., Pillay, N., 2015. A genetic algorithm selection perturbative hyper-heuristic for solving the school timetabling problem. ORiON 31, 39–60.

Ratle, A., 1998. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation, in: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, Springer-Verlag, London, UK. pp. 87–96.

Riley, J., Ciesielski, V., 2010. Fitness landscape analysis for evolutionary non-photorealistic rendering, in: Proceedings of IEEE World Congress on Computational Intelligence, Barcelona.

Rintanen, J., ludwigs-universitat Freiburg, A., 2004. Phase transitions in classical planning: an experimental study, in: In Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference, AAAI Press. pp. 710–719.

Saviniec, L., Santos, M.O., Costa, A.M., 2018. Parallel local search algorithms for high school timetabling problems. European Journal of Operational Research 265, 81 – 98.

Schaerf, A., 1999. A survey of automated timetabling. Artificial Intelligence Review 13, 87–127.

Shaowei, W., Qiuping, Z., 2007. Fitness Landscape Analysis for Optimum Multiuser Detection Problem. Journal of Natural Sciences 12, 1073–1076.

Shen, L., He, J., 2010. A mixed strategy for evolutionary programming based on local fitness landscape, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, pp. 1 –8. doi:10.1109/CEC.2010.5586414.

Sheskin, D.J., 2003. Handbook of parametric and nonparametric statistical procedures. crc Press.

Skoullis, V.I., Tassopoulos, I.X., Beligiannis, G.N., 2017. Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm. Applied Soft Computing 52, 277 – 289.

Slany, K., Sekanina, L., 2007. Fitness landscape analysis and image

filter evolution using functional-level cgp, in: Proceedings of the 10th European conference on Genetic programming, pp. 311–320.

Song, T., Liu, S., Tang, X., Peng, X., Chen, M., 2018. An iterated local search algorithm for the university course timetabling problem. Applied Soft Computing 68, 597 – 608.

Stadler, P., 1995. Towards a theory of landscapes. Complex Systems and Binary Networks , 78–163.

Stadler, P., 1996. Landscapes and their correlation functions. Journal of Mathematical Chemistry 20, 1–45.

Stadler, P.F., Schnabl, W., 1992. The landscape of the traveling salesman problem. Physics Letters A 161, 337 – 344. doi:10.1016/0375-9601(92)90557-3.

Sutton, A.M., Howe, A.E., Whitley, L.D., 2009. Estimating bounds on expected plateau size in maxsat problems, in: Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics, Springer. pp. 31–45.

Sutton, A.M., Howe, A.E., Whitley, L.D., 2010. Directed plateau search for max-k-sat, in: Proceedings of the Third Annual Symposium on Combinatorial Search, pp. 90–97.

Suzuki, H., Iwasa, Y., 1997. Ga performance in a babel-like fitness landscape, in: Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on, pp. 357–366.

Tang, X.L., Yang, S.H., 2013. A improved bus timetable scheduling model using quantum genetic algorithm based on penalty strategy, in: Sustainable Development of Urban Infrastructure, Trans Tech Publications Ltd. pp. 1406–1409.

Tavares, J., Pereira, F.B., , Costa, E., 2008. Multidimensional Knapsack Problem: A Fitness Landscape Analysis. IEEE Transactions on Systems, Man, and Cybernetics -Part B 38, 604–616.

Tavares, J., Pereira, F.B., Costa, E., 2006. The role of representation on the multidimensional knapsack problem by means of fitness landscape analysis, in: Proceedings of IEEE Congress on Evolutionary Computation Sheraton, Vancouver.

Tayarani-N., M., Prugel-Bennett, A., 2014. On the landscape of combinatorial optimization problems. IEEE Transactions on Evolutionary Computation 18, 420–434.

Tayarani-N, M.H., 2020. Novel operators for quantum evolutionary algorithm in solving timetabling problem. Evolutionary Intelligence , 1–25.

Tayarani, N, M.H., Akbarzadeh. T, M.R., 2008. A cellular structure and diversity preserving operator in quantum evolutionary algorithms, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 2665–2670. doi:10.1109/CEC.2008.4631156.

Tayarani-N, M.H., Akbarzadeh-T, M.R., 2014. Improvement of the performance of the quantum-inspired evolutionary algorithms: structures, population, operators. Evolutionary Intelligence 7, 219–239.

Tayarani-N., M.H., Akbarzadeh-T., M.R., 2014. Magnetic-inspired optimization algorithms: Operators and structures. Swarm and Evolutionary Computation 19, 82 – 101. doi:https://doi.org/10.1016/j.swevo.2014.06.004.

Tayarani-N, M.H., Prügel-Bennett, A., 2015. Anatomy of the fitness landscape for dense graph-colouring problem. Swarm and Evolutionary Computation 22, 47–65.

Tayarani-N., M.H., Prügel-Bennett, A., 2015. Quadratic assignment problem: a landscape analysis. Evolutionary Intelligence 8, 165–184.

Tayarani-N., M.H., Prugel-Bennett, A., 2016. An analysis of the fitness landscape of travelling salesman problem. Evolutionary Computation 24, 347–384.

Valouxis, C., Housos, E., 2003. Constraint programming approach for school timetabling. Computers and Operations Research 30, 1555 – 1572. Part Special Issue: Analytic Hierarchy Process.

Vanneschi, L., Tomassini, M., Collard, P., Verel, S., Pirola, Y., Mauri, G., 2007. A comprehensive view of fitness landscapes with neutrality and fitness clouds, in: Genetic Programming. Springer Berlin - Heidelberg. volume 4445 of *Lecture Notes in Computer Science*, pp. 241–250.

Verel, S., Collard, P., Tomassini, M., Vanneschi, L., 2007. Fitness landscape of the cellular automata majority problem: View from the "olympus". Theor. Comput. Sci. 378, 54–77.

Vérel, S., Collard, P., Tomassini, M., Vanneschi, L., 2008. Neutral fitness landscape in the cellular automata majority problem. CoRR abs/0803.4240.

Wang, Y., Feng, X.Y., Huang, Y.X., Pu, D.B., Zhou, W.G., Liang, Y.C., Zhou, C.G., 2007. A novel quantum swarm evolutionary algorithm and its applications. Neurocomputing 70, 633 – 640. doi:https://doi.org/10.1016/j.neucom.2006.10.001. advanced Neurocomputing Theory and Methodology.

Watson, J.P., 2010. An introduction to fitness landscape analysis and cost models for local search. HANDBOOK OF METAHEURISTICS 146, 599–623.

Weinberger, E.D., 1990. Correlated and uncorrelated fitness landscapes and how to tell the difference. Biol. Cyber. 63, 325–336.

Weixiong, Zhang, 2004. Configuration landscape analysis and backbone guided local search.: Part i: Satisfiability and maximum satisfiability. Artificial Intelligence 158, 1 – 26. doi:10.1016/j.artint.2004.04.001.

Whitley, D., Chicano, F., 2012. Quasi elementary landscapes and superpositions of elementary landscapes, in: Proceedings of the 6th international conference on Learning and Intelligent Optimization, Springer-Verlag, Berlin, Heidelberg. pp. 277–291.

Whitley, D., Ochoa, G., 2011. Partial neighborhoods of the traveling salesman problem, in: GECCO'11, pp. 529–536.

Whitley, D., Sutton, A.M., Howe, A.E., 2008. Understanding elementary landscapes, in: GECCO'08, pp. 585–592.

Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biometrics bulletin 1, 80–83.

Wright, S., 1932. The roles of mutation, inbreeding, crossbreeding, and selection in evolution, in: Proceedings of 6th Congress of Genetics, ACM Press. p. 365.

Wu, Y., McCall, J., Corne, D., 2011. Fitness landscape analysis of bayesian network structure learning, in: Evolutionary Computation (CEC), 2011 IEEE Congress on, pp. 981 –988. doi:10.1109/CEC.2011.5949724.

Yoshizawa, H., Hashimoto, S., 2000. Landscape analyses and global search of knapsack problems, in: Systems, Man, and Cybernetics, 2000 IEEE International Conference on, pp. 2311 –2315 vol.3. doi:10.1109/ICSMC.2000.886461.

YuZheng, Jingfa Liu, 2011. A novel quantum-inspired genetic algorithm for a weekly university scheduling optimization, in: International Conference on Information Science and Technology, pp. 373–376.

Zhang, W., 2004. Configuration landscape analysis and backbone guided local search: part i: Satisfiability and maximum satisfiability. Artif. Intell. 158, 1–26. doi:10.1016/j.artint.2004.04.001.

Zhang, W., Rangan, A., Looks, M., 2003. Backbone guided local search for maximum satisfiability, in: Proc. of the 18th Intern. Joint Conference on Artifical Intelligence, pp. 1179–84.

Zhao, F., Xue, F., Yang, G., Ma, W., Zhang, C., Song, H., 2019. A fitness landscape analysis for the no-wait flow shop scheduling problem with factorial representation. IEEE Access 7, 21032–21047. doi:10.1109/ACCESS.2019.2896355.

Zheng, Y., Liu, J., Geng, W., Yang, J., 2009. Quantum-inspired genetic evolutionary algorithm for course timetabling, in: 2009 Third International Conference on Genetic and Evolutionary Computing, pp. 750–753.

Zhou, J., Yin, M., 2010. Phase transitions of plan modification in conformant planning. CoRR abs/1012.2713.

Mohammad- H. Tayarani- N. received his Ph.D. degree from the University of Southampton, Southampton, U.K, in 2013. Then he worked as research fellow at the University of Birmingham, Birmingham, UK and University of Glasgow, Glasgow, UK. He currently holds a fellowship at the university of Hertfordshire, Hatfield, UK. His main research interests include evolutionary algorithms, machine learning, and fractal image

compression.