7

Quantum-Inspired Differential Evolutionary Algorithm for Permutative Scheduling Problems

Tianmin Zheng[1] and Mitsuo Yamashiro[2]
[1]*SoftAgency Co., Ltd*
[2]*Ashikaga Institute of Technology*
Japan

1. Introduction

In the world of combinatorial optimization, a wide range of combinatorial problems exist for which the solution is permutative, such as knapsack problem, traveling salesman problem (TSP), vehicle routine problem (VRP), quadratic assignment problem (QAP), dynamic pick-and-place (DPP) model of placement sequence and magazine assignment in robots and various types of scheduling problems. The objectives for these problems are usually to find the best sequence to realize the optimal, for example, to minimize the maximum completion time of jobs on machines or to find the shortest routing between several cities. These problems are very different from the continuous space problem because we are interested in sequence such as [1 2 3 4 5] which is permutative in nature. The solutions include a permutation-based sequence as well as the fitness. It is not feasible to solve this kind of problem using the approaches that solve only continuous problems.

As an important part of the permutation-based combinatorial optimization problems, the permutative scheduling problems (PSP) account for a large proportion of the production scheduling which is the core content of advanced manufacturing system. Among them, the flow shop scheduling problem (FSP) and job shop scheduling problem (JSP) may be the best known permutation-based scheduling problems. Both of FSP and JSP have earned a reputation for being a typical strongly NP-complete combinatorial optimization problem (Garey, *et al*., 1976) and have been studied by many workers due to their importance both in academic and engineering fields.

By now, the meta-heuristic algorithms achieve global or sub-optimal optima within acceptable time range are most popular for dealing with the permutation-based scheduling optimization problem like flow shop and job shop scheduling. These approaches are initiated from a set of solutions and try to improve these solutions by using some strategies or rules. The meta-heuristics for PSP include genetic algorithm (GA) (Reeves & Yamada, 1998; Goncalves, *et al*., 2005), immune algorithm (IA) (Doyen, *et al*., 2003; Xu & Li, 2007), tabu search (TS) (Nowicki & Smutnicki, 1996; Pezzella & Merelli, 2000), simulated annealing (SA) (Hisao, *et al*., 1995), ant colony optimization (ACO) (Ying & Liao, 2004; Zhang, *et al*., 2006), particle swarm optimization (PSO) (Tasgetiren, *et al*., 2004; Liao, *et al*., 2007; Xia & Wu, 2006), local search (Stützle, 1998), iterated greedy algorithm (Rubén & Stützle, 2007), differential evolution (Pan, *et al*., 2008), and other hybrid approaches (Zheng & Wang, 2003; Qian, *et al*., 2008; Hasan, *et al*., 2009). We should notice that these meta-heuristics can obtain satisfactory solutions, while

require more computation time and vary dramatically according to their structure and parameters. Recently, Han and Kim (2000, 2002, 2004) proposed some quantum-inspired evolutionary algorithms (QEAs) for the knapsack problem. However, due to its encoding and decoding scheme, the QEA can't directly be applied to permutation-based scheduling problems and the research of production scheduling problems based on QEA is just at beginning. Research of PFSP for minimizing the makespan of jobs based on QEA is first proposed by Wang, *et al*. (2005a, 2005b) and he made the simulations and proved that the QEA has better performance than NEH algorithm. Quite recently, Gu, *et al*. (2008) proposed a quantum genetic based scheduling algorithm for stochastic flow shop scheduling problem with the random breakdown and Niu, *et al*. (2009) put forward a quantum-inspired immune algorithm for hybrid flow shop with makespan criterion.

In our study, a novel quantum-inspired evolutionary algorithm called quantum-inspired differential evolutionary algorithm (QDEA) is applied to deal with the FSP and JSP. This chapter is divided into the following sections: section 2 presents the two different shop scheduling problems; in section 3, the basic quantum-inspired evolutionary algorithm is introduced and we put forward the algorithm framework based on QEA for solving the permutation-based scheduling problem. Then, each part of proposed algorithm framework is implemented by developing the novel QDEA which will be presented in section 4. In section 5, we make the simulation and comparisons of proposed QDEA with other algorithms for FSP and JSP. Finally, section 5 concludes the research.

2. Permutative scheduling problems

A flow shop is characterized by continuous and uninterrupted flow of jobs through multiple machines in series and the solution for FSP is the processing sequence of jobs. A FSP containing the same processing sequence of jobs for all machines is called as permutation FSP (PFSP). In the permutation FSP with J jobs and M machines, each job is to be sequentially processed on machine j = 1,2,…,n. At any time, each machine can process at most one job and each job can be processed on at most one machine. The sequence in which the jobs are to be processed is the same for each machine. Here we suppose $\pi = \{J_1, J_2,\ldots,J_n\}$ to be any a processing sequence of all jobs and suppose $c(J_i, k)$ and $t(J_i, k)$ to be the completion time and the processing time of job J_i on machine k, respectively. After initializing $c(J_1, 1) = t(J_1, 1)$, the mathematical formulae for the permutation FSP can be described as follows:

$$c(J_1, k) = c(J_1, k - 1) + t(J_1, k),\ k=2,\ldots,m \tag{1}$$

$$c(J_i, 1) = c(J_{i-1}, 1) + t(J_1, 1),\ i=2,\ldots,n \tag{2}$$

$$c(J_i, k) = max\{c(J_{i-1}, k), c(J_i, k-1)\} + t(J_i, k),\ i=2,\ldots,n,\ k=2,\ldots,m \tag{3}$$

so the scheduling objective such as minimizing the maximum completion time (makespan) which is most widely adopted can be described as:

$$C_{max} = c(J_n, m) \tag{4}$$

The PFSP with the makespan criterion is to find the permutation π^* in the set of all permutations $\prod$ satisfies the following criterion:

$$C_{max}(\pi^*) \le C_{max}(\pi)\ \ \forall\ \pi \in \prod \tag{5}$$

For a deterministic $n \times m$ JSP, the n job $J = \{J_1, J_2, \ldots J_n\}$ must be processed exactly once on each of m machines $M=\{M_1, M_2, \ldots M_n\}$. The processing of a job J_i on one machine M_j is called an operation $O_{i,j}$, and each opeation mush have an integral processing time $p_{i,j}$ ($p_{i,j} > 0$). There are two important constraints make the JSP different from other scheduing problems like PFSP: the operation precedence constraint and the machine processing constraint. The operation constraint means that once order of operations of job is fixed then the processing of an operation cannot be interrupted and concurrent; and the machine constraint is that only a single job can be processed at the same time on the same machine. We denote $C_{i,j}$ as the completion time for operation of job J_i on machine M_j, so the value $C_{i,j} = C_{i,k} + p_{i,j}$ is a completion time of $O_{i,j}$ in relation of which $O_{i,k}$ precedes to $O_{i,j}$ in processing order. For deterministic JSP, the $p_{i,j}$ is pre-set and the goal of scheduling in this study is to find the completion time $C_{i,j}$ for all $O_{i,j}$ to minimize the value of $C_{max} = Max(C_{i,k} + p_{i,j})$, in which the $O_{i,k}$ precedes to $O_{i,j}$ and C_{max} stands for the time used in completing all operations required.

3. Quantum-inspired evolutionary framework for permutative optimization

3.1 The basic quantum-inspired evolutionary algorithm

The quantum-inspired evolutionary algorithm (QEA) is based on the concept and principles of quantum computing, such as the quantum bit and the superposition of states. QEA can explore the search space with a smaller number of individuals and exploit the search space for a global solution within a short span of time. However, QEA is not a quantum algorithm, but a novel evolutionary algorithm (EA). Like any other EAs, QEA is also characterized by the representation of the individual, the evaluation function, and the population diversity. Inspired by the concept of quantum computing, QEA is designed with a novel Q-bit representation, a Q-gate as a variation operator, and an observation process based on Q-bits. QEA uses a novel Q-bit representation which is a kind of probabilistic representation. A Q-bit may be in the "1" state, in the "0" state, or in a linear superposition of two states. A Q-bit individual as a string of Q-bits is defined as:

$$q = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_m \end{bmatrix} \tag{6}$$

where $|a_i|+|\beta_i|=1$, $i=1,2,\ldots,m$ (Han & Kim, 2000, 2002, 2004). If there is, for instance, a two Q-bits system with two pairs of amplitudes such as

$$q = \begin{bmatrix} -1/\sqrt{2} & 1/2 \\ 1/\sqrt{2} & -\sqrt{3}/2 \end{bmatrix} \tag{7}$$

then the states of the system can be represented as

$$-\frac{1}{2\sqrt{2}}|00\rangle + \frac{\sqrt{3}}{2\sqrt{2}}|01\rangle + \frac{1}{2\sqrt{2}}|10\rangle - \frac{\sqrt{3}}{2\sqrt{2}}|11\rangle \tag{8}$$

The above result means that the probabilities to represent the states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are 1/8, 3/8, 1/8, 3/8 respectively. By consequence, the two Q-bits system contains the information of four states. Evolutionary computing with Q-bit representation can provide a better characteristic of population diversity than other representations, since it can represent

linear superposition of states probabilistically. Only one Q-bit individual is enough to represent four states, but in binary representation at least four strings (00), (01), (10), (11) are needed. By observing the states of Q-bit, the Q-bit will collapse into $|0\rangle$ or $|1\rangle$. Then, a quantum chromosome with a length of 2 will become a binary string by a certain measurement method and we can use the binary string to solve the problem required.
The basic QEA includes 4 main steps (Han & Kim, 2000, 2002): initialization, observation, evaluation and updating. The procedure of basic QEA is described in the Figure 1:

```
procedure of QEA{
        t ← 0
        initialize Q(t)
        observe Q(t) and produce P(t)
        evaluate P(t)
        store the best solution b among P(t)
        do{
            t ← t + 1
            observe Q(t -1) and produce P(t)
            evaluate P(t)
            update Q(t)
            store the best solution b among P(t)
        }while(t< MAX_GEN))
}
```

Fig. 1. The pseudo code of basic QEA

When we adopt the QEA to deal with some problems, we firstly initialize the quantum population Q0 = [q0,1, q0,2,...,q0,n], where n is population size and the individual q0,i = [q0,i,1, q0,i,2,...,q0,i,m] is the quantum chromosome represented by two-state Q-bits, where m is the dimension of the problem. Then the step of observation makes binary solutions in P(t) by observing the states of Q(t), where P(t) = [xt,1, xt,2,...,xt,n]. One binary solution xt,j, j=1,2,…,n is a binary string of length, which is formed by selecting either 0 or 1 for each bit using the probability amplitudes. The procedure of evaluation is similar to other EAs by which each binary solution xt,j is evaluated to give a measure of its fitness and the optimum individual b will be stored. In this step of updating, Q-bit individuals in Q(t) are updated by applying quantum rotating gate defined as a variation operator of QEA, the following rotation gate is usually used as a basic Q-gate in QEA:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = U(\theta_i)\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{9}$$

$\theta_i = s(\alpha_i, \beta_i)\Delta\theta_i$, where $\Delta\theta_i$ is the value of the rotation angle and $s(\alpha_i, \beta_i)$ is the rotation direction.
As a conclusion, we can see in the basic QEA, the population provides plenty of diversity even in the small population and it can be easily mixed with other algorithms. In the QEA, the observation and updating are the core of the evolution. The observation operation determines the solution for the specific problem and updating leads the search towards the optimal. In this chapter, we propose a common algorithm framework for the permutative scheduling based on QEA and give an implement to this algorithm framework for solving FSP and JSP in the next section.

3.2 Quantum-inspired evolutionary framework for permutative scheduling problem

This study adopts the QEA to solve the permutative scheduling problem. As for the evolution-based algorithm like QEA, the population of individuals needs to be initialized according to the mechanism of QEA firstly, and effective updating operator also should be adopted to perform the evolution. Then, since the quantum chromosomes can not be used to represent the job permutations directly, a conversion rule is necessary to determine the representation of solutions which is permutative. At last, in order to improve the solution obtained through global search performed by QEA, some neighborhood based search can be performed on the permutative solution to get satisfactory results. The algorithm framework for permutation-based scheduling problems is shown in Figure 2. We will make a brief discussion about the four important parts shown in Figure 2 before implement the algorithm.

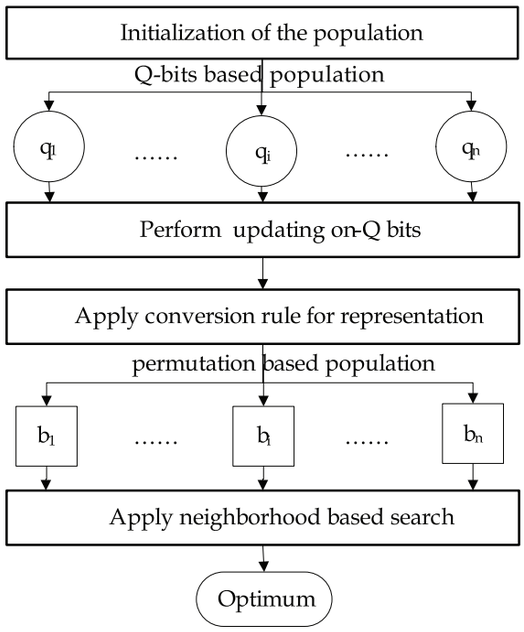


Fig. 2. The algorithm framework for permutation-based scheduling problems

1. The encoding scheme for Q-bits based population
   About how to encode the chromosome composed of several Q-bit, the equation 6 is most widely adopted for solving the scheduling problems (Wang, *et al*., 2005a, 2005b; Gu, *et al*., 2008; Niu, *et al*., 2009). In these researches, only one string of the probability amplitude is used to make the solution and the opposite one is just ignored. While, when we update the quantum individual, according to the normalization condition, these two probability amplitude strings should both be operated. If we just consider one string of probability amplitudes, the other one is a kind of waste. This study will introduce an encoding scheme by which we can make full use of quantum probability information and meet the normalization condition without any unnecessary operation.
2. The updating strategy for quantum evolution
   The updating strategy is the core of the QEA, and Han and Kim (2000, 2002, 2004) use the lookup table to perform the updating of quantum gate. In the lookup table, the value and direction of the rotating angle are determined by making the comparisons between the

current solution and the global best solution. We notice the change of rotating angle is a constant for all the problems, so it is easy to fall into local optimal since different problems has different characteristic. Also, the lookup operation costs excessive search time on the lookup table. The lookup table and its variants are also adopted by the researches (Wang, *et al*., 2005a, 2005b; Gu, *et al*., 2008; Niu, *et al*., 2009). This study adopts a new updating strategy which is more effective to replace the lookup table.

3. The conversion rule for representation of solution
   The decoding scheme is based on the encoding scheme and the specific problem. In the researches (Wang, *et al*., 2005a, 2005b; Gu, *et al*., 2008), the quantum chromosome is converted to job sequence for FSP by using the random key representation proposed by Bean (1994). In their approaches, the quantum chromosome is converted to the binary chromosome firstly, then to the decimal chromosome, and to the job order at last. Although this approach can convert the representation of quantum chromosome into job order, however, with the increasing of the problem scale, the length of corresponding quantum chromosome and the binary chromosome increases rapidly. In this study, we will introduce a simple but efficient way for conversion.
4. The neighbourhood based search for exploitation on job sequence
   Two schedules are neighbours if one can be obtained through a well-defined modification of the other. Neighbourhood search methods provide good solutions and offer possibilities to be enhanced when combined with other heuristics. These techniques continue to add small changes (perturbations) and evaluate schedules until there is no improvement in the objective function. Popular techniques that belong to this family include the tabu search (TS), simulated annealing (SA), genetic algorithm (GA), and so on. At each iteration, these procedures perform search within the neighbourhood and evaluate the various neighbouring solutions. The procedure accepts or rejects a solution as the next schedule based on a given acceptance-rejection criterion. This study will also adopt the neighbourhood based search to improve the solution quality.

Through this algorithm framework, the quantum chromosomes are converted into the job sequence and the algorithm can be designed for dealing with various types of permutation-based scheduling problem. The parts 2 and 4 can be easily replaced since we can use different updating strategies and neighbourhood based search for evolution. Also, the part 3 is problem-dependent and should be adjusted according to different scheduling problems. In the following section, we will propose a novel approach called QDEA by implementing each part of this algorithm framework for the PFSP and JSP both of which are the typical permutation-based production scheduling problem, and give the main procedure of proposed algorithm.

4. Proposed QDEA for permutative scheduling problem

In this section, based on the algorithm framework proposed above, we will implement each part shown in Figure 2 and give the description of proposed QDEA in detail.

4.1 The implement of the QDEA

4.1.1 The initialization of the population

According to the principles of the basic QEA, each Q-bit has two probability amplitudes, so the quantum chromosome consists of two strings of probability amplitude shown in equation 6. In this study, we suppose the quantum chromosome to be represented as:

$$q = [\theta_1 \quad \theta_2 \quad \ldots \quad \theta_n] \tag{10}$$

where θ_i is the quantum rotating angle with the range of $[0, \pi/2]$. We operate on the rotating angle and update it with the updating operation. In the decoding process, the solutions are converted according to the observation method proposed below.

Equation 6 has been adopted to solve the PFSP by Wang, *et al*. (2005a, 2005b), the stochastic FSP by Gu, *et al*. (2008) and hybrid FSP by Niu, *et al*. (2009). Although it is easy to understand and has been widely adopted for dealing with other problems, however, this kind of encoding scheme is not quite effective for the updating operation practiced by the quantum gate. Since the updating operation is the key of the QEA and the essence of the updating procedure is to influence the probability amplitude by changing the value of rotating angle, so we can directly practice on the quantum chromosomes represented in rotating angle. This provides a more effective way to deal with the Q-bits. Also, we can simplify the operations performed on the quantum chromosomes by using only one variable. Therefore, we consider that using the rotation angle to encode the quantum chromosome outperforms that of probability amplitude.

4.1.2 Differential evolution for updating

The differential evolution (DE) is a kind of population based stochastic optimization algorithm proposed by Storn and Price (1997, 1999), and also it's a kind of evolutionary algorithm which is similar to genetic algorithm. The DE adopts the real number encoding scheme, mutation, crossover and selection operation based on differential vectors and has excellent ability of overall search ability.

As with all other evolution based optimization algorithms, DE works with a population of solutions, not with a single solution for the optimization problem. Population x of generation g contains n solution vectors called individuals of the population and each vector represents potential solution for the problem. The population is often initialized by seeding it with random values within the given boundary constraints.

Suppose the population $Q_g = [q_{1,g}, q_{2,g}, \ldots, q_{n,g}]$ (n is the population scale, g is the current evolutionary generation), individual $q_{i,g} = [\theta_{i,1,g}, \theta_{i,2,g}, \ldots, \theta_{i,m,g}]$ (m is the dimension of the problem, $i \in [1, n]$). Suppose $v_{i,g+1}$ is the corresponding individual obtained by practicing the mutation operator on individual $q_{i,g}$, and one type of mutation operators works as:

$$v_{i,g+1} = q_{r1,g} + F(q_{r2,g} - q_{r3,g}) \tag{11}$$

where $r1, r2, r3 \in [1, n]$ and $r1 \neq r2 \neq r3 \neq i$; $q_{r1,g}$ is called father basic vector, $(q_{r2,g} - q_{r3,g})$ is called father differential vector; F is a real number and constant factor which controls the amplification of the differential variation.

In order to increase the diversity of the parameter vectors, we also use the $u_{i,j,g+1}$ ($j \in [1, m]$) vector which is obtained by practicing kinds of crossover operation between $q_{i,g}$ and mutative individual $v_{i,g+1}$ obtained by equation 11. The bin crossover we will use in this study is showed in equation 12:

$$u_{i,j,g+1} \begin{cases} v_{i,j,g+1} & for \quad rand < CR \quad or \quad j = Jrnd \\ \theta_{i,j,g} & otherwise \end{cases} \tag{12}$$

where CR is the crossover factor and $Jrnd$ is chosen randomly from the interval $[1, m]$.

We adopt the differential operation to perform the updating of the quantum chromosomes which is an innovation in this study. Since the quantum chromosomes are encoded in quantum angle, so the differential operations are directly practiced on the quantum angle and can provide the updating with excellent overall search ability and diversity.

4.1.3 The representation of permutation-based solution

For the decoding process of the quantum chromosome, since the solution to permutative scheduling problems is the order of all the elements (jobs for FSP, operations for JSP), therefore, we should convert the quantum chromosome encoded in rotating angle to job or operation sequences. In the decoding scheme adopted by Wang, *et al*. (2005a, 2005b) and Gu, *et al*. (2008), the representation needs several conversions (Q-bit chromosome → binary chromosome → decimal chromosome → job order) and the computation is complicated when the problem scale becomes larger. We put forward a simple but efficient strategy for conversion, which is also an innovation in this study.

- **Initialization**
  1. Obtain quantum chromosome $q_i = [\theta_{i,1}, \theta_{i,2},\ldots,\theta_{i,n}]$ from the Q-bit based population; Calculate $temp_i = [\cos\theta_{i,1}, \cos\theta_{i,2},\ldots,\cos\theta_{i,n}]$ and initiate two arrays *first*() and *last*().
- **Convertion**
  2. Generate a random number η between [0,1] and compare it with $\cos\theta_{i,e}$ where $e \in [1, n]$. If $\cos\theta_{i,e} > \eta$, put e into *first*() , else put e into *last*(). Repeat until all Q-bits in q_i are operated.
  3. Combine these two arrays *first*() and *last*() to one array *permutation*(), the *element* in *permutation*() is the permutative sequence for solution.
- **FSP representation**
  4. For PFSP, *permutation*() is the final solution.
- **JSP representation**

5. For JSP, get a element p from *permutation*() and perform the $code(i) = \text{mod}(p, m) + 1$, where the "mod" is an operator of calculating the reminder p being divided by m and m is the machine number. Repeat until all p in permutation() are operated. The *code*() is the solution for JSP.

Fig. 3. The procedure of converting mechanism for solution representation

For the solution reprentation of permutative problems, we define the rotating angles of element 1,2,…n are $[\theta_1, \theta_2,\ldots,\theta_n]$, that the probability amplitude of element i is $[\cos\theta_i, \sin\theta_i]$, then determine the permutative sequence according to the steps shown in Figure 3.

An example of JSP (suppose machine constrains to be [2-1; 1-2; 2-1] for 3 jobs) for converting mechanism is shown in Figure 4. For example, we have q_i = [0.87, 0.68, 0.15, 0.42, 1.38, 1.09], so the $temp_i$ = [0.64, 0.77, 0.98, 0.91, 0.18, 0.46]. Then generate η as 0.76 which is larger than $\cos\theta_{i,1}$, so we put '1' into *last*(). We continue generate η as 0.37 which is smaller than $\cos\theta_{i,2}$, so we put '2' into *first*(). After we operated on all 6 Q-bits, we have *first*() = [2 3 6] and *last*() = [1 4 5]. By combining these two arrays, we have *permutation*() = [2 3 6 1 4 5]. By applying the mod(p, m) + 1 operation, the JSP code becomes [3 1 1 2 3 2]. So each job number occurs m

times in the chromosome, and by reading the array *code*() from 1 to $n \times m$, the i-th occurrence of a job number refers to the i-th operation in the technological sequence of this job. Through the operation-based representation, any permutation can be decoded to a feasible schedule for JSP. In Figure 4, by scanning the elements in the job shop code, we can get the final schedule of $\{O_{312}, O_{112}, O_{121}, O_{211}, O_{321}, O_{221}\}$, where O_{ijk} means the j-th operation of job i is processed on the machine k.

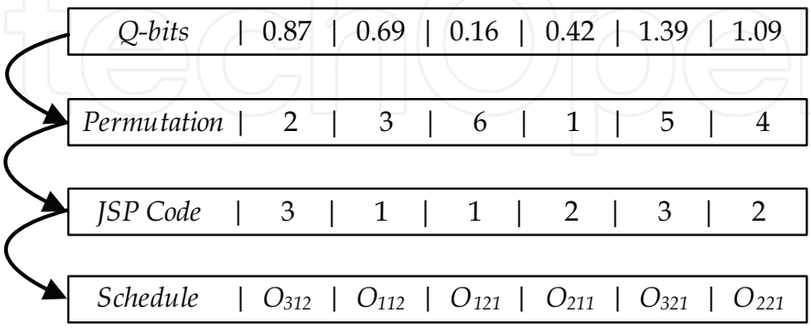


Fig. 4. The example of converting mechanism for JSP

Here, suppose we have a scheduling problem with the scale of 30 jobs and 10 machines and we can make a comparison like this way: for PFSP, since $2^4 < 30 < 2^5$, so the length of the quantum chromosome should be 30×5 = 150 at least by the method proposed by Wang, *et al*. (2005a, 2005b) and Gu, *et al*. (2008), and three conversions are needed to get the final solution. While, by our method, we just need a quantum chromosome with length of 30 and practice the conversion only once; for JSP, since the solution is the operation sequence, so the length of the quantum chromosome should be 30×10×5 = 1500 for these approaches, and five conversions are needed to get the operation. While, by our method, we just need a quantum chromosome with length of 300 and practice the conversion only three times. Thus, the representation of the solution we proposed simplifies the decoding procedure for the quantum chromsomes greatly and can provide a more effective way to deal with permutative scheduling problems.

4.1.4 Hybrid QDEA with local search scheme

By adopting the proposed converting mechanism, the Q-bits based population can be converted to permutative-based solution for scheduling effectively, so various types of neighborhood based search can be easily embedded to develop effective hybrid algorithms. For the permutative-based optimization problems, the *Insert*, *Interchange*, *Swap*, *2-opt* and *Or-opt* neighborhoods are often adopted. According to the analysis from Schiavinotto and Stützle (2007), for the permutative-based search landscape, using *Insert* at most n-1 times, one solution can transit to any other solution. Compared with other commonly used neighborhoods, the diameter of *Insert* is one of the shortest ones, so here we adopt the following local search scheme based on *Insert* neighborhood to perform thorough exploitation in the promising permutation-based solutions.

We obtain the global best chromosome *Best_g* and suppose n to be the element (job for PFSP and operation for JSP) number for a special problem. The pseudo code of the local search adopted is given as follows:

- **Initialization**
  1. Initialize counter t = 0, calculating makespan of Best_g as M(Best_g).
- **Perform the search**
  2. **Remove.** Generate i randomly and remove i-th job in *Best_g* and obtain a partial sequence *temp.*
  3. **Insert.** Insert the removed job into the best position $j(j \neq i)$ in *temp* and calculate $M(Best\_g)$.
- **Stopping condition check**
  4. If $M(temp) > M(Best\_g)$ and $t < n^{1/2}$, then $t \leftarrow t + 1$ and go to step 2.
- **Update**
  5. If M(temp) < M(Best_g), update Best_g = temp.

Fig. 5. The procedure of local search

Through the local search by using *Insert* neighborhood, we can obtain two job sequences stand for the one before the operation and the one after the operation, the better one is saved for the next generation iteration. It is should be noticed that the local search is directly applied on the job permutation, not on the quantum chromosomes. So after the whole local search completes, the corresponding Q-bit chromosome should be repaired since this quantum chromosome will be used to perform the next updating by DE and should match the permutation results obtained by the local search. The repair operation is simple since we just need to exchange the corresponding position of the Q-bits.

element	**1**	**2**	**3**	**4**	**5**	**6**
q_i	0.87	0.69	0.16	0.42	1.39	1.09
p_i	2	3	6	1	4	5
$q_{i'}$	0.87	**1.09**	0.16	0.42	1.39	**0.69**
$p_{i'}$	2	5	6	1	4	3

Table 1. An example of repair for local search

For example, before the local search, the Q-bit chromosome q_i and corresponding element permutation p_i obtained by conversion are shown in Table 1. After the local search, the element permutation $p_{i'}$ becomes [2 5 6 1 4 3] in which the '3' and '5' change the 2-th and 6-th position according to the *Insert* neighborhood, so we make the repair by changing the corresponding 2-th and 6-th Q-bits in q_i and obtain the $q_{i'}$= [0.87, 1.09, 0.16, 0.42, 1.39, 0.69]. The $q_{i'}$ will be adopted to perform the updating by DE strategy in the next iteration.

4.2 The main procedure of QDEA

To implement the common algorithm framework introduced in section 3.2, we adopt the DE strategy to perform the updating of quantum gate and introduce an effective converting mechanism for representing the permutative solution. In this way, we propose the QDEA for PFSP and JSP. Also, we develop the hybrid QDEA (HQDEA) by embedding the local search scheme to perform the neighborhood based search. The main procedure of QDEA (along with HQDEA) is as follows:

- **Initialize control parameters**
  1. Set the value of control parameters for DE and the lower and upper of angle for the QEA. Set maximum evolution generation *iter* and initialize iteration counter $t = 0$.
- **Initialize the population**
  2. Determine initial population $Pop_0 = [chrom_{0,1}, chrom_{0,2},..., chrom_{0,n}]$, where $chrom_{0,i} = [\theta_{0,1}, \theta_{0,2},..., \theta_{0,m}]$, n is the population scale, m is the number of jobs and $\theta_{0,j} \in [0, \pi / 2]$. In this step, the initial quantum chromosomes are generated randomly.
- **Make the solution**
  3. Adopt the converting mechanism to make the solution for permutation-based problem from the Q-bits based population.
- **Evaluate the population**
  4. Obtain objective values by evaluating Pop_0, store the best one into $Best_0$; store the best individual into $Best\_\theta$ and best element sequence into $Best\_g$. In this step, we perform the evaluation operation based on element sequence and get the objective values by calculating the permutative solution.
- **Perform the evolution**
  5. Update the Pop_{t-1} to Pop_t by using PSO strategy. In order to provide the search with excellent diversity and guarantees the normalization of Q-bits, we need to make sure that the individuals in Pop_t should be in the range between the lower(0) and upper(0.5π), so if any individual is out of this range, it should be revised by using $\theta_t =$ lower + rand × (upper − lower).
  6. Adopt the converting mechanism to make the solution for permutation-based problem from the Q-bits based population.
  7. Evaluate Pop_t and get objective values, compare with the corresponding solution in $Best_{t-1}$ and store the better one to $Best_t$. Update the $Best\_\theta$ and $Best\_g$.
  8. Practice the local search operations on $Best\_g$ by using the *Insert* operator and make the repair of the Q-bit chromosomes.
- **Stopping condition check**
  9. If the stopping condition $t >$ *iter* is met or the optimum is found, output the optimum; else $t \leftarrow t + 1$ and go to step 5.

Fig. 6. The main procedure of proposed QDEA

When perform the evolution, the updating of PSO is practiced on the quantum chromosome encoded in rotating angle and the local search is practiced on permutative element sequence, respectively. It should be noticed that for the updating performed by the PSO, we first need to save the $Best\_\theta$ as θ_gbest along with the v_t and θ_pbest_t for each Q-bit before we perform the updating next time.

5. Simulations and comparisons

The validation of proposed QDEA and HQDEA is conducted on the two demanding problems of PFSP and JSP which both are the typical permutative scheduling problems. Each experiment is conducted in two phases. The first phase is to introduce the benchmark and measure adopted in the simulations and to experimentally obtain the operating parameters for QDEA. The second phase is the comparison of the QDEA and HQDEA with other established approaches reported in the literature.

5.1 Simulations and comparisons on PFSP

5.1.1 Preparations for simulation

To test the performance of the proposed QDEA and HQDEA for PFSP, computational simulation is carried out with some well-studied benchmarks. In this study, four problem sets are selected. The first eight problems are called car1, car2 through car8 by Carlier (1978). The second 21 problems are called rec01, rec03 through rec41 by Reeves (1995). The third 110 problems are from Taillard (1993) and the last problem sets are called DMU problems from Demirkol, *et al*. (1998). Thus far these problems have been used as benchmarks for study with different methods by many researchers.

In order to make comparisons by using different methods, we adopt the following measures widely used in other literatures:

1. *RE*: the Relative Error of the average solution after we run the algorithm n times. The BRE and ARE stand for the best and average relative percentage error to the *Opt*, where *Opt* denotes the optimal solution value known thus far. The BRE and ARE can be calculated as equation 13 (the *S* is short for solution):

$$BRE(ARE) = \sum_{r=1}^{n}\left(\frac{S_{best,average} - Opt}{Opt} \times 100\right) \times \frac{1}{n}(\%) \tag{13}$$

2. *APRD*: the percentage relative deviation (PRD) of a best solution value P_{best} found by an algorithm from the optimal solution P_{opt} can be calculated as:

$$PRD = (P_{best} - P_{opt}) \,/\, P_{opt} \times 100\% \tag{14}$$

   and *APRD* is the average *PRD* values for a set of instances.

3. *ARPI:* the relative percentage increase (PRI) is defined as: suppose the best solution value found by an algorithm A denoted as P_A, and the best solution value found by n algorithms (include algorithm A) denoted as P_k, k = 1,2,...,n, then the PRI can be calculated as:

$$PRI = (P_A - \min(P_k, k = 1,2,...,n)) \,/\, \min(P_k, k = 1,2,...,n) \times 100\% \tag{15}$$

   and *APRI* is the average *PRI* values for a set of instances.

Parameter selection may influence the quality of the results. For the differential evolutionary strategy, two parameters should be set properly, one is the crossover factor *CR* and the other is the weight factor *F*. In this study, we will make a comparison between the 25 combination of $CR \in \{0.1,0.3,0.5,0.7,0.9\}$ and $F \in \{0.1,0.3,0.5,0.7,0.9\}$. In order to determine the appropriate values of parameters, a preliminary simulation is performed on 8 selected instances from the Rec benchmark problem set. For the 25 group data, we run 20 times for each group and calculate the average solution. We find that with the different group of parameters, the results vary dramatically and what we want to do is to choose a group of *F*/*CR* with the good and stable performance for all of 8 problems. The computational results indicate that the best solution is obtained at values F = 0.1 and CR = 0.9, so these values are adopted for all further experiments in this study.

At the same time, the different evolutionary strategy leads to different performance. For the PFSP, we make the simulation by comparing the performance of different evolutionary strategies proposed by Storn and Price (1999) in the form of $DE/x/y/z$, where x determine the vector to be operated is randomly generated or the best one, i.e. rand/best; y means the

number of the differential variables, i.e. 1/2 and here z stands for the *exp* or *bin* crossover. In this preliminary simulation, we want to make comparisons by using 10 evolutionary strategies: $DE1,2 \rightarrow DE/best/1/z$; $DE3,4 \rightarrow DE/rand/1/z$; $DE5,6 \rightarrow DE/rand\text{-}to\text{-}best/1/z$; $DE7,8 \rightarrow DE/rand/2/z$ and $DE9,10 \rightarrow DE/best/2/z$, where z has two values for crossover. After we made the simulation, we notice that for the F/CR = 0.1/0.9, the *DE7* strategy with the *bin* crossover provides the best performance for different benchmark problems, so we choose the *DE7* strategy to perform the updating of the quantum chromosomes.

In the following QDEA algorithm, we set F = 0.1 and CR = 0.9, differential evolutionary strategy by *DE7* and maximum iterative time I_{max} = 500. Based on the preliminary simulations, these values will give high probability to obtain better solutions. To test the performance of the proposed QDEA and HQDEA, computational simulation is carried out with the selected benchmark problems. The Visual C++ 6.0 is used to program the algorithm and all the computations are conducted on a Celeron 1.59 GHZ with 512 MB memory. We run the algorithm 20 times for each problem and use the statistical results for discussion.

Since many meta-heuristics and hybrid meta-heuristics have been adopted for solving the PFSP in literature, we want to make some comparisons to demonstrate the superiority of our QDEA. In this section, two types of recent and effective particle swarm optimization algorithm will be compared with QDEA firstly. Then we make a comparison between QDEA and quantum-inspired genetic algorithm (QGA) for both single objective and multi-objective PFSP to show the effectiveness of the proposed encoding scheme, converting mechanism and updating operation. And we also want to show our hybrid QDEA can obtain better optimization results than other hybrid algorithms.

5.1.2 Comparison of QDEA with CPSO and DPSO

Particle swarm optimization has two versions: continuous PSO (CPSO) and discrete PSO (DPSO). In this section, continuous PSO proposed by Tasgetiren, *et al*. (2004), discrete PSO proposed by Liao, *et al*. (2007) are compared with QDEA. In the CPSO proposed by Tasgetiren, *et al*., they proposed a SPV rule for solution representation and adopted the local search and mutation operation to avoid the premature convergence; in DPSO, the evolution is performed by defining the discrete particle and velocity trail, and the construction of a particle sequence is proposed for the PFSP. The experiments are conducted on the DMU problems from Demirkol, *et al*. (1998) (available from http://cobweb.ecn.purdue.edu/~uzsoy2/benchmark/fcmax.txt) in accordance with these two algorithms. In these 40 benchmark instances, eight combinations with number of machines m = 15, 20 and number of jobs n = 20, 30, 40, 50 are randomly generated and the best upper bounds for these instances are also provided by authors. We use the QDEA to minimize the makespan of jobs and the simulation results are given in Table 2.

In Table 2, the results are shown in the form of APRD. To be fair, the APRD of CPSO, DPSO and QDEA are calculated within the same running time for each problem sets (Liao, *et al*. 2007): [1.25s, 1.55s, 3.30, 3.95s, 6.40s, 7.60s, 11.00s, 12.90s] by setting the population size *Np* and number of iterations *iter*. From the results, we can see that DPSO is superior to CPSO for most of problem sets both in average (Avg) and minimum (Min) APRD, and our QDEA is much better than DPSO by comparing the APRD. Especially for Min APRD, the proposed QDEA has overwhelming superiority than DPSO which means our algorithm can obtain minimal makespan than DPSO. This simulation clearly demonstrates the excellent population diversity and unique optimization performance of Q-bits based search.

n×*m*	CPSO by Tasgetiren, *et al*			DPSO by Liao, *et al*			Proposed QDEA		
	Np×*iter*	Avg	Min	*Np*×*iter*	Avg	Min	*Np*×*iter*	Avg	Min
20×15	40×2,500	−6.54	−7.68	100×1,000	−6.47	−7.93	20×2,000	−7.37	−11.65
20×20		−4.93	−6.20		−4.92	−6.20		−6.25	−8.11
30×15	60×2,500	−7.22	−8.75	150×1,000	−7.37	−9.05	30×2,000	−7.96	−11.06
30×20		−5.67	−7.44		−5.79	−7.56		−6.52	−11.18
40×15	80×2,500	−7.80	−9.31	200×1,000	−8.06	−9.74	40×1,000	−8.16	−11.26
40×20		−5.60	−7.39		−5.61	−6.87		−5.43	−7.97
50×15	100×2,500	−6.47	−7.70	250×1,000	−6.71	−7.92	50×1,500	−6.68	−8.95
50×20		−7.23	−8.81		−7.18	−8.43		−7.20	−9.43
Average		−6.40	−7.86		−6.44	−7.86		−6.94	−9.95

Table 2. Results of testing two PSOs and QDEA

5.1.3 Comparison of QDEA with QGA for single objective PFSP

To show the effectiveness of the coding scheme and the updating strategy proposed in this study, we want to compare the QDEA with the quantum-inspired genetic algorithm (QGA) developed by Wang, *et al*. (2005a) based on the Car and Rec benchmark problems (available from http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop1.txt). The QGA adopts the random key to represent the solution and the lookup table to perform the updating which are all discussed in section 3.2. We firstly make the comparison on the single objective of minimizing the makespan and the results are shown in Table 3.

From Table 3, we can see the QDEA is overwhelming over the QGA for all the Car and Rec problems and even the AREs of QDEA are better than the BREs of QGA for most of instances. Since we do not perform the local search on the permutative solutions and the differences between QDEA and QGA are the coding scheme and the updating strategy only, so we can conclude the coding scheme and the updating strategy proposed in this study is more suitable for dealing with the permutation-based scheduling problems like the PFSP. The allowed running times (in second) of QDEA are also listed in the Table 3 for reference.

5.1.4 Comparison of QDEA with QGA for multi-objective PFSP

In the real world manufacturing environment, practical problems often involve multiple objectives that need to be considered concurrently, both from a process planning and a scheduling perspective. To apply the proposed QDEA to multi-objective PFSP and compare the performance with QGA (Wang, *et al*., 2005a), we conduct an experiment inspired from the research made by Sridhar and Rajendran (1996). They developed a genetic algorithm for the PFSP with triple objectives—makespan, total flow time, and total machine idle time. A DELTA operator is used to determine whether the parents should be replaced by the children and a single solution with equal weights for the three objectives is finally produced. Based on equation 1 to equation 3, the total flow time C_{sum} and total machine idle time I_{sum} can be obtained by calculating the $C_{sum} = \sum c(J_i, m)$ and $I_{sum} = \{ c(J_1, k-1) + \sum\{\max\{ c(J_i, k-1) - c(J_{i-1}, k) , 0\}\} | k = 2,...,m\}$ for i from 1 to n. Both evaluation operation and the updating operation of QDEA and QGA should be modified for dealing with the multi-objective problems. To establish a measure for these triple objectives, we include a modified version

P	N,M	C_{max}*	QGA		QDEA		
			BRE	ARE	BRE	ARE	Time
Car1	11,5	7,038	0.00	0.00	0.00	0.00	0.41
Car2	13,4	7,166	0.00	1.90	0.00	0.00	0.47
Car3	12,5	7,312	1.09	1.65	0.00	0.07	0.44
Car4	14,4	8,003	0.00	0.06	0.00	0.00	0.50
Car5	10,6	7,720	0.00	0.11	0.00	0.10	0.38
Car6	8,9	8,505	0.00	0.19	0.00	0.15	0.34
Car7	7,7	6,590	0.00	0.00	0.00	0.00	0.28
Car8	8,8	8,366	0.00	0.03	0.00	0.00	0.33
Rec01	20,5	1,247	2.81	6.79	0.00	0.47	0.87
Rec03	20,5	1,109	0.45	3.87	0.00	0.46	0.89
Rec05	20,5	1,242	2.25	3.00	0.24	0.45	0.88
Rec07	20,10	1,566	1.05	4.67	0.00	1.20	1.25
Rec09	20,10	1,537	4.03	6.40	0.65	2.70	1.23
Rec11	20,10	1,431	6.08	8.79	0.42	2.24	1.22
Rec13	20,15	1,930	5.08	7.98	1.64	3.17	1.55
Rec15	20,15	1,950	3.49	5.93	1.17	3.11	1.56
Rec17	20,15	1,902	6.51	9.10	2.35	4.20	1.55
Rec19	30,10	2,093	7.98	9.80	2.79	5.28	2.39
Rec21	30,10	2,017	6.94	10.05	2.01	4.19	2.38
Rec23	30,10	2,011	9.10	10.55	3.67	4.99	2.39
Rec25	30,15	2,513	7.16	10.06	3.18	5.10	3.13
Rec27	30,15	2,373	7.63	11.05	3.33	4.65	3.14
Rec29	30,15	2,287	12.42	14.06	2.61	6.01	3.15
Rec31	50,10	3,045	9.82	12.68	5.21	7.10	5.90
Rec33	50,10	3,114	6.20	9.54	1.10	3.48	5.81
Rec35	50,10	3,277	4.21	6.52	0.89	3.18	5.91
Rec37	75,20	4,951	15.54	17.49	8.15	9.10	7.80
Rec39	75,20	5,087	13.50	15.49	5.90	7.43	7.79
Rec41	75,20	4,960	16.92	18.84	8.10	9.02	7.79
AVE			5.18	7.12	1.84	3.03	2.47

Table 3. The comparisons between QDEA and QGA for minimizing makespan

of evaluation operation by assigning suitable weights to the three objectives in order to obtain a single solution. In this study, the weights are determined in accordance with Franminan, *et al*. (2002) to balance the effect of magnitude and the single solution is calculated as $1/3\times C_{max}\times n/2+1/3\times C_{sum}+1/3\times I_{max}\times n/10$. When to update the quantum

chromosome, the DELTA (Sridhar and Rajendran 1996) operator is used to make the comparison between the current solution (denoted as CS_t) and the previous best solution (denoted as PS_{t-1}) for t-th iteration of evolution. By evaluating CS_t and PS_{t-1}, the values of two makespans of $C_{t,max}$ and $C_{t-1,max}$, total flow times of $C_{t,sum}$ and $C_{t-1,sum}$, and total idle times of $I_{t,sum}$ and $I_{t-1,sum}$ can be obtained and the DELTA is defined as follows:

$$\begin{aligned} DELTA = {} & w_1(C_{t,max} - C_{t-1,max}) / \min(C_{t,max}, C_{t-1,max}) + w_2(C_{t,sum} - C_{t-1,sum}) / \min(C_{t,sum}, \\ & C_{t-1,sum}) + w_3(I_{t,sum} - I_{t-1,sum}) / \min(I_{t,sum}, I_{t-1,sum}) \end{aligned} \quad (16)$$

where $w_1 = w_2 = w_3 = 1/3$. So if $DELTA>0$, it indicates CS_t is better than PS_{t-1} and we update the corresponding Q-bits; otherwise, we keep PS_{t-1} as the best solution for this iteration.
We program both of the QDEA and QGA by using above evaluation and updating strategy. To have a fair comparison, we run these two algorithms in the same computer by using the same soft of Visual C++ 6.0 and within the same running time. The 90 Taillard's problems (Taillard, 1993) are adopted to perform the simulation and the PRI shown in equation 15 is used for the performance measure. The comparison results of the proposed QDEA and QGA along with the computation time for each problem set are summarized in Table 4.

$n\times m$	QDEA			QGA			computation time (s)
	C_{max}	C_{sum}	I_{sum}	C_{max}	C_{sum}	I_{sum}	
20×5	0.00	0.00	0.00	1.39	1.56	3.12	0.81
20×10	0.00	0.00	0.00	1.32	1.73	2.45	1.25
20×20	0.00	0.41	0.00	0.87	0.00	0.56	1.80
50×5	0.00	0.00	0.00	3.14	1.96	4.01	3.25
50×10	0.00	0.00	0.00	2.56	1.21	2.16	4.85
50×20	0.00	0.23	0.32	1.78	0.00	0.00	5.70
100×5	0.00	0.00	0.00	2.16	2.13	5.34	6.16
100×10	0.00	0.00	0.00	3.18	1.56	4.24	7.79
100×20	0.00	0.00	0.00	3.23	2.15	6.12	9.18
Average	0.00	0.07	0.04	2.18	1.37	3.11	4.53

Table 4. Results of testing two algorithms for multi-objective PFSP

We give the average value of PRI (APRI) for the three objectives. From the Table 4, it can be observed that for the multi-objective FSP, the proposed QDEA performes as well as the single one. Compared to the QGA, the QDEA can obtain the best value in general. Although for the criterion of total flowtime and total idle time, the QDEA is slightly inferior to QGA for several problem sets, the QDEA performs best among most of problems for these two criterions and all of problems for minimizing the makespan, which also demonstrates that the proposed QDEA has the prospects in the real world production scheduling applications.

5.1.5 Comparison of HQDEA with HGA, HQGA and HDE

To show the effectiveness of proposed hybrid QDEA embedded with the local search, we carry on comparisons with some popular hybrid algorithms. In this section, we make the comparisons between HQDEA and the hybrid genetic algorithm (HGA) proposed by Zheng

& Wang (2003), the hybrid quantum-inspired evolutionary algorithm (HQGA) proposed by Wang, *et al*. (2005b) and the hybrid differential evolution (HDE) algorithm proposed by Qian, *et al*. (2008) based on Car and Rec problems. HGA uses multi-crossover operators

P	N,M	C_{max}*	HGA		HQGA[a]		HDE		HQDEA	
			BRE	ARE	BRE	ARE	BRE	ARE	BRE	ARE
Car1	11,5	7,038	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Car2	13,4	7,166	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Car3	12,5	7,312	0.000	1.504	0.000	0.000	0.000	0.000	0.000	0.000
Car4	14,4	8,003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Car5	10,6	7,720	0.000	0.938	0.000	0.000	0.000	0.000	0.000	0.000
Car6	8,9	8,505	0.000	2.132	0.000	0.000	0.000	0.000	0.000	0.000
Car7	7,7	6,590	0.000	1.003	0.000	0.000	0.000	0.000	0.000	0.000
Car8	8,8	8,366	0.000	1.281	0.000	0.000	0.000	0.000	0.000	0.000
Rec01	20,5	1,247	0.160	0.192	0.000	0.140	0.000	0.144	0.000	0.112
Rec03	20,5	1,109	0.000	0.271	0.000	0.170	0.000	0.000	0.000	0.009
Rec05	20,5	1,242	0.242	0.628	0.240	0.340	0.242	0.242	0.242	0.242
Rec07	20,1	1,566	0.115	1.149	0.000	1.020	0.000	0.230	0.000	0.000
Rec09	20,1	1,537	0.605	1.627	0.000	0.640	0.000	0.000	0.000	0.000
Rec11	20,1	1,431	0.000	1.532	0.000	0.670	0.000	0.000	0.000	0.000
Rec13	20,1	1,930	0.415	1.974	0.160	1.070	0.104	0.301	0.104	0.225
Rec15	20,1	1,950	0.615	2.385	0.050	0.970	0.000	0.308	0.000	0.158
Rec17	20,1	1,902	1.840	2.482	0.630	1.680	0.000	1.178	0.000	0.126
Rec19	30,1	2,093	1.113	2.676	0.290	1.430	0.287	0.559	0.287	0.435
Rec21	30,1	2,017	1.522	1.636	1.440	1.630	0.198	1.413	0.149	1.041
Rec23	30,1	2,011	0.497	2.188	0.500	1.200	0.448	0.482	0.348	0.597
Rec25	30,1	2,513	1.922	2.706	0.770	1.870	0.478	1.492	0.119	0.995
Rec27	301,	2,373	1.551	2.318	0.970	1.830	0.843	1.285	0.253	0.954
Rec29	30,1	2,287	2.610	3.629	0.350	1.970	0.306	0.791	0.000	0.824
Rec31	50,1	3,045	1.156	2.759	1.050	2.500	0.296	0.824	0.263	0.565
Rec33	50,1	3,114	0.450	1.188	0.830	0.910	0.000	0.434	0.000	0.297
Rec35	50,1	3,277	0.000	0.131	0.000	0.150	0.000	0.000	0.000	0.000
Rec37	75,2	4,951	4.312	5.096	2.520	4.330	1.818	2.727	1.717	2.771
Rec39	75,2	5,087	2.597	3.205	1.630	2.710	0.983	1.541	0.845	1.485
Rec41	75,2	4,960	4.133	5.599	3.130	4.150	1.673	2.649	1.190	1.965
AVE			0.892	1.801	0.502	1.082	0.265	0.572	0.175	0.428

Table 5. Results of testing three hybrid algorithms and HQDEA

[a] In HQGA, the results are accurate to the second decimal place.

acting on the divided subpopulations and replaces the classical mutation by SA; in HQGA, the Q-bit representation is converted to random key representation which genetic operation are practiced on, and a permutation-based genetic algorithm is also applied after the solutions are constructed; as for the HDE, it not only applies the parallel evolution mechanism of DE to perform effective exploration, but also adopts problem-dependent local search to perform exploitation. The statistic performances of the four algorithms for the criterion of minimizing makespan are given in Table 5.

From the Table 5, we can see that for the Car problems with small scale, the HGA, HQGA, HDE and HQDEA all can find the optimum; for the Rec problems with relatively large scale, HQDEA also provide us with better performance which means the BREs and AREs are much smaller than that of the HGA and better than or equal to HQGA and HDE for all the problems. Also, we can notice that for the problem Rec01, Rec03, Rec07, Rec09, Rec11, Rec15, Rec17, Rec29, Rec33 and Rec35, the HQDEA has found the best solution known up to now. So the proposed HQDEA is a novel and effective approach for the PFSP.

5.2 Simulations and comparisons on JSP

5.2.1 Preparations for simulation

To test the performance of the proposed HQDEA for JSP, computational simulation is carried out with some well-studied benchmarks. In this study, 43 benchmarks (available from http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html) are selected. The first 3 problems are called FT06, FT10 and FT20. The other 40 problems are called LA01, LA02 through LA40.

In order to evaluate the performance of different algorithms for JSP, the following four measures will be introduced:

1. Minimum makespan (MS): it is used for evaluating quality of solution. For the JSP, the minimum makespan a certain algorithm can achieve is usually adopted to prove the search ability of this algorithm.
2. Average convergence generation (CG): at each running of HQDEA, the optimal or the sub-optimal solution will be found after a number of generations. For several simulation replications, the number may be different, so this metric also reflects the average convergence speed of an algorithm.
3. Average compulation time (CT): the average computation time (in second) for an algorithm to find the optimal (or sub-optimal). Since different approaches run in different machines, the comparisons based on the CPU times might not seem to be meaningful. While, when make the comparison on the same PC, this metric can be used to show the effectiveness of an algorithm.
4. Relative error (RE): same to the definition of equation 13 in section 5.1.1

The parameters are set same to the preliminary simulation results given in section 5.1.1.

5.2.2 Comparison of QDEA with QGA

Firstly, we want to compare the proposed QDEA (without the local search operation) with the QGA developed by Wang, *et al*. (2005a) based on the three FT benchmark problems to show the effectiveness of the coding scheme and the updating strategy for JSP. We program both of these two algorithms with the same decoding procedure proposed in this study and run them on the same PC, we set the population size to be the number of job for each problem and the iteration time I_{max} to be 300 for both two algorithms, each instance runs 20 times and the results are shown in Table 6.

	FT06		FT10		FT20	
	QDEA	QGA	QDEA	QGA	QDEA	QGA
CG	187	204	192	178	217	254
MS	55	55	980	1125	1276	1465
RE	1.455	2.909	13.694	23.226	18.318	29.528
CT	0.480	1.208	4.518	19.888	4.478	19.795

Table 6. The results of QDEA and QGA on FT problems

From Table 6, we can see the QDEA is overwhelming over the QGA for all the three problems, especially for the FT10 and FT20, QDEA obtained much better makespan within about only 1/4 of the running time of QGA. Since we do not perform the local search on the permutative solutions and the differences between QDEA and QGA are the coding scheme and updating strategy only, so we can conclude that the coding scheme and updating strategy proposed in this study is also suitable for dealing with the JSP.

5.2.3 Comparison of HQDEA with HQGA

In order to check the effectiveness of proposed HQDEA with local search for JSP, we run the algorithm by combining the QDEA and local search and make the comparison with the HQGA proposed by Wang, *et al*. (2005b). The results are shown in Table 7. From Table 7, we notice the HQDEA can find all the optimums for three FT problems within 200 generations, while the HQGA can not achieve the optimum of FT10 and FT20 even spend more time and run more generations. The comparisons of relative error also shows the effectiveness of the propose HQDEA.

	FT06		FT10		FT20	
	HQDEA	HQGA	HQDEA	HQGA	HQDEA	HQGA
CG	1.1	9	132	221	148	276
MS	55	55	930	937	1165	1178
RE	0	0	1.785	4.430	0.961	3.867
CT	0.220	0.356	140.344	193.061	170.993	190.728

Table 7. The results of HQDEA and HQGA on FT problems

5.2.4 Comparisons between HQDEA with other approaches

To further show the effectiveness of HQDEA, we carry on some comparisons with other popular algorithms include the hybrid genetic algorithm (HGA) proposed by Goncalves, *et al*. (2005), memetic algorithm (MA) by Hasan, *et al*. (2009), tabu search (TSSB) by Pezzella & Merelli (2000), hybrid particle swarm optimization (HPSO) by Xia & Wu (2006) based on 40 LA benchmarks. We run the HQDEA using the same settings in section 5.2.2, and the results are shown in Table 8. In Table 8, the 'BKS' refers to the best solution found by now for each LA problems, 'average gap' is calculated as: $(MS-BKS)/BKS\times 100\%$ and 'No. of BKS obtained' means how many BKS can be found by an algorithm.

P	N,M	BKS	HGA	MA	TSSB	HPSO	HQDEA		
							MS	RE	CG
LA01	10,5	666	666	666	666	666	666	0.000	2
LA02	10,5	655	655	655	655	655	655	0.217	15
LA03	10,5	597	597	597	597	597	597	0.271	21
LA04	10,5	590	590	590	590	590	590	0.119	10
LA05	10,5	593	593	593	593	593	593	0.000	1
LA06	15,5	926	926	926	926	926	926	0.000	1
LA07	15,5	890	890	890	890	890	890	0.000	1
LA08	15,5	863	863	863	863	863	863	0.000	1
LA09	15,5	951	951	951	951	951	951	0.000	1
LA10	15,5	958	958	958	958	958	958	0.000	1
LA11	20,5	1222	1222	1222	1222	1222	1222	0.000	1
LA12	20,5	1039	1039	1039	1039	1039	1039	0.000	1
LA13	20,5	1150	1150	1150	1150	1150	1150	0.000	1
LA14	20,5	1292	1292	1292	1292	1292	1292	0.000	1
LA15	20,5	1207	1207	1207	1207	1207	1207	0.000	1
LA16	10,10	945	945	945	945	945	945	0.836	88
LA17	10,10	784	784	784	784	784	784	0.045	51
LA18	10,10	848	848	848	848	848	848	0.259	85
LA19	10,10	842	842	842	842	842	842	0.481	104
LA20	10,10	902	907	907	902	902	902	0.527	117
LA21	15,10	1046	1046	1079	1046	1047	1046	0.738	112
LA22	15,10	927	935	960	927	927	927	0.912	121
LA23	15,10	1032	1032	1032	1032	1032	1032	0.000	3
LA24	15,10	935	953	959	938	938	935	0.892	212
LA25	15,10	977	986	991	979	977	977	1.111	241
LA26	20,10	1218	1218	1218	1218	1218	1218	0.000	6
LA27	20,10	1235	1256	1286	1235	1236	1235	1.109	243
LA28	20,10	1216	1232	1286	1216	1216	1216	0.354	113
LA29	20,10	1157	1196	1221	1168	1164	1161	1.256	221
LA30	20,10	1355	1355	1355	1355	1355	1355	0.000	2
LA31	30,10	1784	1784	1784	1784	1784	1784	0.000	1
LA32	30,10	1850	1850	1850	1850	1850	1850	0.000	1
LA33	30,10	1719	1719	1719	1719	1719	1719	0.000	1
LA34	30,10	1721	1721	1721	1721	1721	1721	0.000	1
LA35	30,10	1888	1888	1888	1888	1888	1888	0.000	1
LA36	15,15	1268	1278	1307	1268	1269	1268	1.086	212
LA37	15,15	1397	1408	1442	1411	1401	1401	1.611	265
LA38	15,15	1196	1219	1266	1201	1208	1201	1.896	216
LA39	15,15	1233	1246	1252	1240	1240	1238	1.123	234
LA40	15,15	1222	1241	1252	1233	1226	1224	1.011	247
Average gap (%)			0.4190	1.0708	0.1091	0.0842	0.0404		
No. of BKS obtained			28	27	33	31	35		

Table 8. The comparisons between HQDEA and other algorithms

From Table 8, we can see for the LA01 to LA20, these 5 algorithms all can find optimum, especially for most problems, the HQDEA just needs to run the algorithm once for searching the optimization space. For the difficult problems with the middle and large scale, some problems remain to be unsolved in the provided evolution iteration. While, the HQDEA has obtained the 35 optimum out of 40 problems and achieved the minimum average gap of 0.0404 among these 5 algorithms. All these demonstrate that the HQDEA we proposed is a novel, effective and robust approach for JSP. The relative error s for HQDEA are also list in Table 8 for reference.

6. Conclusions and future research

In this study, we proposed an improved quantum-inspired evolutionary algorithm called quantum-inspired differential evolutionary algorithm (QDEA) for solving the flow shop scheduling and job shop scheduling problems with permutation-based solutions. Based on the QEA, we proposed a simple converting mechanism to determine permutative sequence based on quantum chromosome encoded in the form of rotating angle. Then we studied the applications of the QDEA by adopting the differential evolution strategy to perform the updating of quantum gate and local search to perform thorough exploitation in the promising permutative solutions. We adopt this novel QDEA to deal with the single objective and the multi-objective permutation FSP and to minimize the makespan of JSP. Compared to other algorithms, the simulation results demonstrated the effectiveness of our algorithm. For the PFSP, the QDEA performed better than two PSO based algorithms (Tasgetiren, *et al*., 2004; Liao, *et al*., 2007) and the QGA (Wang, *et al*., 2005a) for both of single objective and multi-objective problem; the proposed hybrid QDEA also provides better results than the hybrid algorithms include HGA (Zheng & Wang, 2003), HQGA (Wang, *et al*., 2005b) and HDE (Qian, *et al*., 2008); for the JSP, we also obtained satisfactory results by comparing QDEA with QGA and HQDEA with other state-of-the-art approaches (Goncalves, *et al*., 2005, Hasan, *et al*., 2009, Pezzella & Merelli, 2000, Xia & Wu, 2006). All these show the excellent diversity of the Q-bits based search and the effectiveness of the local search.

This study has made a step towards establishing an efficient heuristic for the permutation-based scheduling problems based on the quantum-inspired evolutionary algorithm. In this study, we propose a common algorithm framework for permutation-based scheduling problems, and the QDEA is in fact one implementation to this algorithm framework. As for the future research work, we can extend this study in the following ways. For the part 2 of algorithm framework, the parameter settings for the differential evolution strategy are worth examining in detail firstly. Then, developing new hybrid strategy by combining Q-bit based search and other evolution based methods to improve the performance also makes a great sense. For the part 4 of algorithm framework, we can use other strategies like variable neighbourhood search (VNS) to perform the neighbourhood based search and check the performance. At last, for the application of this research, the proposed QDEA approach can be extended to deal with flow and job shop scheduling problems with different constraints and performance criteria; also we can apply this new method to other permutation-based shop scheduling problems such as open shop scheduling problem (OSP) and make comparisons with other algorithms.

7. References

Bean J C. (1994) Genetics and random keys for sequencing and optimization. *ORSA Journal of computing*, 6(2), 154-160.

Bin Qian, Ling Wang, Rong Hu, Wan-Liang Wang, De-Xian Huang and Xiong Wang. (2008). A hybrid differential evolution method for permutation flow-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 38(5-6), 757-777.

Carlier J (1978) Ordonnancements a Contraintes Disjonctives. *Recherche Operationelle /Operations Research*, 12(4), 333-350.

Ching-Jong Liao,Chao-Tang Tseng and Pin Luarn. 2007. A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Research*, 34(10), 3099-3111.

D. Zheng and L.Wang (2003). An Effective Hybrid Heuristic for Flow Shop Scheduling. The *International Journal of Advanced Manufacturing Technology*. 21(1), 38-44.

Demirkol E, Mehta S, Uzsoy R (1998) Benchmarks for shop scheduling problems. *Eur J Oper Res* 109:137–141.

Doyen A, Engin O, Ozkan C (2003). A new artificial immune system approach to solve permutation flow shop scheduling problems. *Tukish Symposium on Artificial Immune System and Neural Networks* TAINN'03.

Framinan JM, Leisten R, Ruiz-Usano R. (2002) Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation. *European Journal of Operational Research*, 141, 559–69.

Garey M, Johnson D and Sethi R (1976) The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 24(1), 117-129.

Goncalves, J. F., Mendes, J. J. M., and Resende, M. G. C. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1), 77–95.

Han K-H. (2000) Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem. In: *IEEE Proc. Of the 2000 Congress on Evolutionary Computation*，San Diego，USA IEEE Press, July 2000.

Han K-H，Kim J-H. Quantum-inspired Evolutionary Algorithm for a class of Combinatorial Optimization. *IEEE Trans on Evolutionary Computation*, 2002.

Han K-H，Kim J-H. Quantum-inspired Evolutionary Algorithms with a New Termination Criterion H,Gate and Two-Phase Scheme. *IEEE Trans on Evolutionary Computation* 2004.

Hisao Ishibuchi, Shinta Misaki and Hideo Tanaka (1995) Modified simulated annealing algorithms for the flow shop sequencing problem. *European Journal of Operational Research*. 81(2), 388-398.

Jinwei Gu, Xingsheng Gu, Bin Jiao. (2008). A Quantum Genetic Based Scheduling Algorithm for stochastic flow shop scheduling problem with random breakdown. *Proceedings of the 17th World Congress. The International Federation of Automatic Control* Seoul, Korea, July 6-11, 63-68.

Jun Zhang, Xiaomin Hu, X. Tan, J.H. Zhong and Q. Huang. (2006). Implementation of an Ant Colony Optimization technique for job shop scheduling problem. *Transactions of the Institute of Measurement and Control*, 28(1), 93-108.

Kuo-Ching Ying and Ching-Jong Liao (2004) An ant colony system for permutation flow-shop sequencing. *Computers & Operations Research*. 31(5), 791-801.

Nowicki E, Smutnicki C (1996) A fast tabu search algorithm for the permutation flow-shop problem. *Eur J Oper Res* 91,160–175.

Pezzella, F., & Merelli, E. (2000). A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120(2), 297–310.

QK Pan, MF Tasgetiren, YC Liang (2008) A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering*, 55(4), 795-816.

Qun Niu, Taijin Zhou, Shiwei Ma. (2009). A Quantum-Inspired Immune Algorithm for Hybrid Flow Shop with Makespan Criterion. *Journal of Universal Computer Science*, 15(4), 765-785.

Reeves, C R (1995) A genetic Algorithm for Flowshop Sequencing. *Computers and Operations Research*, 22(1), 5-13.

Reeves CR, Yamada T (1998) Genetic algorithms, path relinking and the flowshop sequencing problem. *Evol Comput* 6, 45–60.

Rubén Ruiz, and Thomas Stützle (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*. 177(3), 2033-2049.

S. M. Kamrul Hasan, Ruhul Sarker, Daryl Essam and David Cornforth. (2009). Memetic Algorithms for Solving Job-Shop Scheduling Problems. *Memetic Computing*, 1(1), 69-83.

Schiavinotto T, Stützle T. (2007). A review of metrics on permutations for search landscape analysis. *Computers Operations & Research*, 34(10), 3143–53.

Sridhar J, Rajendran C. (1996). Scheduling in flowshop and cellular manufacturing systems with multiple objectives—a genetic algorithmic approach. *Production Planning and Control*, 7, 374–82.

Storn R, Price K (1997) Differential evolution—a simple evolution strategy for fast optimization. *Dr. Dobb's Journal*, 78, 18–24.

Storn R, Price K. (1999). Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report*, TR-95-012, ICSI.

Stützle, T (1998) Applying iterated local search to the permutation flow shop problem. *Technical report*, AIDA-98-04, FG Intellektik, TU Darmstadt.

Taillard, E (1993) Benchmarks for basic scheduling problems. *European journal of operational research*, 64(2), 278-285.

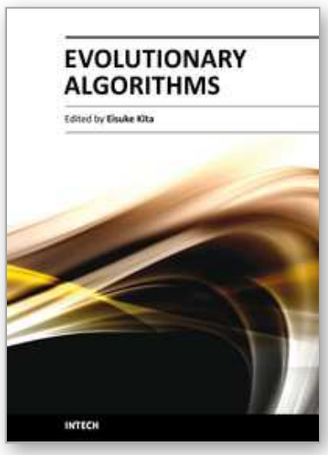
Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G. (2004) Particle swarm optimization algorithm for makespan and maximum lateness minimization in permutation flowshop sequencing problem. In: *Proceedings of the fourth international symposium on intelligent manufacturing systems*, urkey: Sakarya; 431–41.

Xiao-dong Xu & Cong-xin Li. (2007). Research on immune genetic algorithm for solving the job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*. 34, 783–789.

Wang Ling, Wu Hao, and Zheng Da-Zhong (2005a) A quantum-inspired genetic algorithm for scheduling problems. *Lecture Notes in Computer Science*, v 3612, n PART III, Advances in Natural Computation: First International Conference, ICNC 2005. Proceedings, 417-423.

Wang L, Wu H, Tang F and Zheng DZ (2005b) A hybrid quantum-inspired genetic algorithm for flow shop scheduling. *Lecture Notes in Computer Science*, 3645, 636-644.

Wei-jun Xia, Zhi-ming Wu. (2006). A hybrid particle swarm optimization approach for the job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*. 29, 360–366.



Evolutionary Algorithms

Edited by Prof. Eisuke Kita

ISBN 978-953-307-171-8
Hard cover, 584 pages
Publisher InTech
Published online 26, April, 2011
Published in print edition April, 2011

Evolutionary algorithms are successively applied to wide optimization problems in the engineering, marketing, operations research, and social science, such as include scheduling, genetics, material selection, structural design and so on. Apart from mathematical optimization problems, evolutionary algorithms have also been used as an experimental framework within biological evolution and natural selection in the field of artificial life.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen