A Quantum-Inspired Evolutionary Approach for non-Homogeneous Redundancy Allocation in Series-Parallel Multi-State Systems
Yan-Fu Li, Enrico Zio

To cite this version:

HAL Id: hal-01108136
https://hal-supelec.archives-ouvertes.fr/hal-01108136
Submitted on 22 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A Quantum-Inspired Evolutionary Approach for non-Homogeneous Redundancy Allocation in Series-Parallel Multi-State Systems

Yan-Fu Li, Enrico Zio
Chair on Systems Science and the Energetic Challenge, European Foundation for New Energy-Electricite’ de France
Ecole Centrale Paris-Supelec
Paris, France
yanfu.li@ecp.fr, yanfu.li@supelec.fr

Abstract— Redundancy allocation is a family of well-known reliability optimization problems. The non-homogeneous type of redundancy allocation in series-parallel multi-state systems is among the most difficult ones. Evolutionary algorithms (EAs) are frequently applied to solve the problem, mainly due to the huge search space and the non-closed-form system reliability. This work proposes an efficient approach that combines a quantum-inspired evolutionary algorithm (QEA) with a newly designed local search strategy. Different from the existing EAs, it is able to evolve an explicit probabilistic model to explore the search space in an iterative way. The proposed method is tested on two benchmark problems with the comparisons to the published results. The results are promising in terms of both solution quality and computation efficiency.

Keywords--parallel multi-state system, redundancy allocation problem, quantum inspired evolutionary algorithm, local search.

I. INTRODUCTION

Redundancy allocation problem (RAP) is a well-known optimization problem for the design of many industrial systems [1-3]. It aims to maximize system reliability or minimize system cost for given constraints on cost, reliability, weights, etc. RAP is a NP-hard [4] problem of non-linear and combinatorial nature. Most of the existing RAP works are based upon a binary state system model, which assumes that the system and its elements have only two states: perfect functioning and complete failure.

The multi-state system (MSS) model has recently gained increasing popularity for system reliability assessment because it realistically considers more than one intermediate states for the system and its elements, between the two extremes of perfect functioning and complete failure. The MSS version of the RAP has been first investigated in [5], where the universal generating function (UGF) approach [6] was used for reliability computation. Due to the high complexity of this problem, meta-heuristics are mainly used as solution techniques. The existing studies include genetic algorithm (GA) [1, 2, 7], Tabu search (TS) [8, 9], ant colony optimization (ACO) [10, 11], particle swarm optimization (PSO) [12], etc. In these implementations, the RAP is set to obtain the optimal series-parallel MSS (SPMSS) structure that minimizes the system cost while maintaining the system reliability above a predefined level.

There are two kinds of RAPs. The first kind allows only one type of component that can be used in each subsystem, namely the homogeneous RAP. The second kind allows the mixture of components in each subsystem, namely the non-homogeneous RAP.
homogeneous RAP. The latter one is more challenging due to its larger solution space [13].

In this paper we propose a novel quantum-inspired evolutionary algorithm (QEA) to solve the SPMSS RAP of second type. QEA developed by Han and Kim [14] is by far the most promising application of the quantum mechanics concepts [15] onto heuristic optimization. A number of successful applications have been reported across various optimization problems [14, 16]. Based upon the concepts and principles of quantum computing, e.g. quantum bits (Q-bits), quantum gates (Q-gate) and superposition of states [17, 18], QEA is able to automatically achieve a good balance between exploration and exploitation of the solution space, and obtain quality solutions with a small population compared to the conventional evolutionary algorithms (EAs) [19].

The rest of this paper is organized as follows. The formulation of the non-homogeneous SPMSS RAP is presented in Section 2. Section 3 presents the proposed QEA approach including a novel local search (LS) strategy and constraints handling. In Section 4, the effectiveness of the formulation of the non-homogeneous SPMSS RAP is compared to published results. Section 5 concludes the work.

II. FORMULATION OF RAP

A. Definitions and assumptions of multi-state series parallel system

The SPMSS typically consists of \( N \) subsystems connected in series. The \( i \)-th (1 \( \leq i \leq N \)) subsystem has \( n_i \) components connected in parallel, belonging to \( v_i \) versions. The \( j \)-th (1 \( \leq j \leq v_i \)) version component at the \( i \)-th subsystem has \( m_{ij} \) states \( \{0,1, \ldots, m_{ij}\} \), where state \( m_{ij} \) and 0 are perfect functioning and complete failure states, respectively. The \( k \)-th (0 \( \leq k \leq m_{ij} \)) state is characterized by the performance level \( g_{ijk} \) and the state probability \( p_{ijk} \). Additionally, there is a cost \( c_{ij} \) for the \( j \)-th version component at the \( i \)-th subsystem.

The following assumptions are made for SPMSS model:
1. The element states are mutually independent.
2. The mixing of components of different versions is allowed.
3. The state of the system is completely determined by the state of its components.
4. All components are repairable.

B. Formulation of RAP

Let \( x_{ij} \) denote the number of components (integer value) of \( j \)-th version at the \( i \)-th subsystem. The RAP aims to minimize the total system cost \( C = \sum_{i=1}^{N} \sum_{j=1}^{v_i} c_{ij} x_{ij} \) while keeping the system availability \( A \) equal to or above a predefined level \( A_0 \). The formulation is presented as follows:

Minimize: \[ C = \sum_{i=1}^{N} \sum_{j=1}^{v_i} c_{ij} x_{ij} \] (1)

Subject to:
\[ A \geq A_0 \] (2)
\[ \max x_{ij} \geq x_{ij} \geq 0 \] (3)

The second constraint specifies the range for the number of components of each version. The computation of \( C \) is straightforward. To compute \( A \), the UGF approach is typically adopted [1, 7, 8, 12]. More details about this technique can be found in [20]. We present the basic steps in the following.

The UGF of the \( j \)-th (1 \( \leq j \leq v_i \)) version component at the \( i \)-th subsystem is

\[ u_{ij} (z) = \sum_{m_{ij}}^{m_{ij}} \sum_{k}^{p_{ijk} z^{g_{ijk}}} \] (4)

The UGF of the \( i \)-th subsystem is written as

\[ u_i (z) = \prod_{j=1}^{v_i} u_{ij} (z) \] (5)

where the composition operator \( \otimes \) is used to derive the UGF of a subsystem consisting of components connected in parallel. The generic composition operator \( \otimes \) between any two combined components is defined as follows

\[ u_{ij} (z) \otimes u_{ij} (z) = \sum_{m_{ij}}^{m_{ij}} \sum_{k}^{p_{ijk} z^{g_{ijk}}} f^{g_{ijk} z^{g_{ijk}}} \] (6)

where \( f(\cdot) \) is the structure function reflecting the topology of the component combination. For examples, ‘+’ represents the parallel combination and ‘\( \cdot \)’ represents the series combination. More details about the generic composition operator \( \otimes \) can be found in [20].

Based upon (5), the UGF of SPMSS can be written as

\[ u_i (z) = u_1 (z) \otimes u_2 (z) \otimes \ldots \otimes u_N (z) \] (7)

Suppose it has the following expanded form

\[ u_i (z) = \sum_{i=1}^{N} p_{ij} z^{g_{ij}} \] (8)

Given the arbitrary system demand \( W_{it} \) at the \( t \)-th operation time step, the system availability \( A_{it} \) at this time is computed as

\[ A_{it} = \Psi \left( \sum_{i=t}^{N} p_{ij} z^{g_{ij} w_{it}} \right) = \sum_{i=t}^{N} \Psi \left( p_{ij} Z^{g_{ij} w_{it}} \right) \] (9)

where \( \Psi \) is the distributive operator [20] with the following definition

\[ \Psi (p z^{g-w}) = \begin{cases} p, & \text{if } g \geq W \\ 0, & \text{if } g < W \end{cases} \] (10)

For the entire operation period which is divided into \( T \) time steps, the system availability \( A \) is computed as

\[ A = \left( \sum_{t=1}^{T} A_{it} d_{it} \right) / \sum_{t=1}^{T} d_{it} \] (11)

where \( d_{it} \) is the duration of the \( t \)-th time step.

III. QEA APPROACH

A. Solution representation

In analogy to the bit in conventional EA encoding, the Q-bit, i.e. quantum bit [21], serves as the smallest information unit in QEA. Unlike the classical bit, which has to be either
state ‘0’ or state ‘1’, a Q-bit can be ‘0’, ‘1’, or a superposition of both states. Let |0⟩ and |1⟩ denote the two basis states, respectively; the state of one Q-bit $|\psi\rangle$ can be written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$  \hspace{1cm} (12)

where $\alpha$ and $\beta$ are the probability amplitudes and they have to satisfy that

$$|\alpha|^2 + |\beta|^2 = 1$$  \hspace{1cm} (13)

It is noted that $|\alpha|^2$ and $|\beta|^2$ are the probabilities of state ‘0’ and state ‘1’, respectively. A Q-bit individual is a string of $l$ concatenated Q-bits

$$q = [q_1 q_2 ... q_l] = \begin{bmatrix} \alpha_1 & \alpha_2 & ... & \alpha_l \end{bmatrix}$$  \hspace{1cm} (14)

For each Q-bit $q_i$, the condition (13) must be satisfied. To evaluate the fitness of a Q-bit individual $q$, each $q_i$ is first sampled to form a binary bit $b_i \in \{0, 1\}$. This sampling is done according to the probability $|\beta_i|^2$ of state ‘1’. In fact, $q$ defines a probabilistic model

$$p = [|\beta_1|^2 |\beta_2|^2 ... |\beta_l|^2]$$  \hspace{1cm} (15)

This model explicitly describes the probability distribution of the solutions in the search space and is able to sample $2^l$ different binary bit solutions.

### B. QEA procedures

The detailed procedures of QEA are presented as follows.

#### Initialization: set $t$, the generation index, equal to 0 and randomly generate the population $Q_t = \{q^1_i, ..., q^{n_p}_i\}$ (where $n_p$ is the total number of individuals in the population). Each individual $q^i_i$ ($i = 1, ..., n_p$) takes the form as presented in eq. (14) and all its Q-bits $a^i_{ij} = \frac{\alpha^i_{ij}}{\beta^i_{ij}}$ ($j = 1, ..., l$) equal to the value $1/\sqrt{2}$ so that the probabilities of observing |1⟩ and |0⟩ are the same for each Q-bit.

#### Observation: sample a binary population $B_t = \{b^1_i, ..., b^{n_p}_i\}$ from $Q_t$. For each individual $b^i_j$ = $[b^i_{ij}, ..., b^i_{il}]$, each of its element $b^i_{ij}$ is binary and determined by comparing $|\beta^i_{ij}|^2$ with a uniformly distributed random number in the range [0, 1]. If $|\beta^i_{ij}|^2 > \text{rand}[0, 1]$ then $b^i_{ij} = 1$; otherwise $b^i_{ij} = 0$.

#### Evaluation: evaluate each individual in $B_t$ using the fitness or objective function. In this study, the fitness function is a penalized form of the system cost (1). (see after its definition (17))

#### Elitism: create a population of elite solutions $E_t = \{e^1_i, ..., e^{n_p}_i\}$ to store each binary individual $b^i_j$ initially sampled. It is noted that $E_t$ can be divided into a number of equally sized local groups. Within each group, the solutions have the ability to synchronize themselves with the best individual among them, periodically. In addition, all the solutions in $E_t$ are periodically replaced by the best one $e^t_i$ found in the entire $E_t$. More details about the elitism strategy (also named ‘migration’ by Han and Kim) can be found in [14].

#### Set $t = t + 1$.

**Observation**: sample a binary population $B_t$ from $Q_{t-1}$.

**Evaluation**: evaluate each individual in $B_t$, using the fitness or objective function $f(\cdot)$.

**Variation**: update each Q-bit individual using the Q-gate [14], which is the analog to variation operators such as crossover and mutation in classical EA. In QEA, the variation operator is the rotation gate $U(\Delta \theta_j)$.

$$U(\Delta \theta_j) = \begin{bmatrix} \cos(\Delta \theta_j) & -\sin(\Delta \theta_j) \\ \sin(\Delta \theta_j) & \cos(\Delta \theta_j) \end{bmatrix}$$  \hspace{1cm} (15)

where $\Delta \theta_j$ is the rotation angle determining the magnitude and direction of the rotation for the $j$-th Q-bit, and it should be designed in compliance with the application problem. Using this gate, the $j$-th Q-bit of $q^t_i$ is updated as follows,

$$\begin{bmatrix} \alpha^t_{ij} \\ \beta^t_{ij} \end{bmatrix} = U(\Delta \theta_j) \begin{bmatrix} \alpha^{t-1}_{ij} \\ \beta^{t-1}_{ij} \end{bmatrix}$$  \hspace{1cm} (16)

It is seen that this operator should satisfy the normalization condition. Figure 1 shows the polar plot of such updating.

![Figure 1. Polar plot of the rotation gate for updating one Q-bit](image-url)
positive value (or $\delta$) which indicates the increase of probability of sampling state $|1\rangle$; 2) if $q_j^t$ is in the II or IV quadrant, $\Delta\theta_j$ will be a negative value (or $-\delta$) which means an increase of the probability of sampling state $|0\rangle$.

**Table 1. Lookup Table of the Rotation Angle**

<table>
<thead>
<tr>
<th>$f(b^t_j) &gt; f(e^t_j)$</th>
<th>$b^t_j$</th>
<th>$e^t_j$</th>
<th>$\Delta\theta_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>0</td>
<td>1</td>
<td>$\delta$ (Q-bit in I/III quadrant)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>$-\delta$ (Q-bit in II/IV quadrant)</td>
</tr>
<tr>
<td>False</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Elitism:** obtain $E_t$ by assigning to each $e^t_j$ the best individual from the pair $b^t_j$ and $e^{t-1}_j$.

**Termination:** stop the algorithm if the termination criteria are met; otherwise go to Step 6.

In literature there are a number of Q-gates, e.g. NOT gate, controlled NOT gate, or Hadamard gate [21]. The rotation gate is most frequently applied in QEA.

**C. QEA-SPMSS-RAP Approach**

In this Section, the classical QEA is tailored for SPMSS RAP. In particular, it includes the encodings of the solution to RAP, the constraints handling, the novel local search strategy, and the complete procedures of the QEA approach.

**Encodings**

In line with the problem formulation in Section 2.2, a solution to RAP can be represented by a vector $x = [x_{11}, \ldots, x_{1t_p}; \ldots; x_{N1}, \ldots, x_{Nt_p}]$. Because the QEA operation is based upon binary variables, for each integer variable $x_{ij}$ within the range $[0, MAX_{ij}]$ we use its binary equivalent $[b_{ij1}, \ldots, b_{ijn}]$ where $n_{ij} = \lfloor \log_2 MAX_{ij} \rfloor$. Gray coding is used for the decimal-binary conversion, because in this system two successive values are different by only one bit. The Q-bit individual takes the form described in eq. (14) with the length equal to that of its corresponding binary individual.

**Constraints handling**

The penalty function approach is used to handle the constraint in eq. (2). To effectively explore the feasible and infeasible solutions near the border of the feasible area, a penalty approach inspired by the BSS work [22, 23] is used. It is then added onto the original system cost function. The penalized system cost as the following expression,

$$C_p = \begin{cases} 
C, & A \geq A_0 \\
C + d \left(1 + \frac{A_0}{d}\right), & otherwise 
\end{cases} \quad (17)$$

where $C$ is the original system cost presented in eq. (1) and $d$ is a relatively large constant dependent on the specific problem.

**Local search (LS)**

LS has shown to be effective in improving the candidate solutions obtained by the main algorithm for solving RAP [12, 22]. In this study, we design two LS methods alternately applied to each individual in the binary population. They are modified from the LS strategies c) and e) proposed in [12], with the emphasis on exploring less expensive solutions. At each LS operation, only one of the LS methods is performed.
on a randomly selected subsystem \( i \); the other LS method is performed at the next LS operation. The first LS method deletes one component of a randomly selected version \( j_{r_{i}} \) whose \( c_{i_j} > 0 \) (i.e. \( x_{i_j} \leftarrow x_{i_j} - 1 \)); then, it adds one component to a randomly selected version \( j_{r_{i}} \) which is less expensive than version \( j_{r_{i}} \) (i.e. \( x_{i_j} \leftarrow \min ( x_{i_j} + 1, \text{MAX}_{i_j} ) \)). Note that the upper limit \( \text{MAX}_{i_j} \) must be kept. If the component being deleted is of the least expensive version, then no component will be added. For example, if a subsystem consists of one version 1 component version 1 of cost 35 $ and nominal performance 50%, three version 2 components of cost 20 $ and nominal performance 30%, and two version 3 components of cost 15 $ and nominal performance 25%, then a feasible LS operation is to remove one version 2 component and add one version 3 component. Another option is to simply delete one version 1 component.

The second LS method randomly selects one existing component version \( j_{r_{i}} \) to be deleted (i.e. \( x_{i_j} \leftarrow 0 \)), and then randomly select another version \( j_{r_{i}} (r_{i} \neq r_{i}) \) to be increased. The number of components to be added is \( n_{r_{i}} = \text{round} \left( \frac{g_{i_{r_{i}} - m_{i_{r_{i}}}}}{g_{i_{r_{i}}} - m_{i_{r_{i}}}} \right) \), where \( g_{i_{r_{i}} m_{i_{r_{i}}}} \) is the nominal performance of the component version \( j \) at subsystem \( i \). If \( n_{r_{i}} c_{i_{r_{i}} j} < x_{i_j} c_{i_{r_{i}} j} \), then \( x_{i_j} \leftarrow \min ( x_{i_j} + n_{r_{i}} \text{MAX}_{i_{r_{i}}} ) \); otherwise, perform neither the deletion nor the addition. Using the exemplar system above, a feasible LS operation is to remove one version 1 component and add two version 3 components. However, to remove two version 3 components and add one version 1 component is not feasible.

**Overall procedure**

The overall procedure of the proposed optimization approach is represented by the flow chart in Figure 2. Note that \( T_{p} \) denotes the period for LS operation and it starts at the first LS method application.

### IV. EMPIRICAL VALIDATIONS

**A. Experiment design**

To compare with other published algorithms, the proposed QEA approach is tested on two well known benchmark problems. The first problem (P1) consists of four subsystems connected in series [2]. For each subsystem, there are 4 to 6 different component versions available. The availability requirement \( A_{0} \) is set to three different values, namely 0.900, 0.960, 0.990, to create three test cases. The second problem (P2) consists of five subsystems connected in series [7]. For each subsystem, there are 4 to 9 different component versions available. The availability requirement is set to be 0.975, 0.980 and 0.990. The data sets of the two problems can be found in [24]. The upper limit \( \text{MAX}_{i_{r_{i}}} \) is set to be 7, the same to all test cases.

The parameters of QEA approach include population size \( n_{p} \), maximum generation \( g_{\text{max}} \), absolute rotation angle \( \delta_{0} \), penalty constant \( d \), and LS period \( T_{p} \). As QEA typically needs a very small population, we set \( n_{p} = 5 \) for all the experiments. We set \( g_{\text{max}} = 2000 \) for P1 and P2. The value of \( \delta_{0} \) is problem-dependent [14]. To choose an optimal one, in this study we change \( \delta_{0} \) from 0.005\( \pi \) to 0.050\( \pi \) with step size of 0.005\( \pi \), following [16]. For each problem, different \( \delta_{0} \) values are first evaluated on the test case with \( A_{0} = 0.99 \) and, then, the \( \delta_{0} \) value which produces the lowest average cost is used for all the test cases of this problem. In the end, we have \( \delta_{0} = 0.010\pi \) and 0.030\( \pi \) for P1 and P2, respectively. The penalty constant \( d \) needs to have a sufficiently large value [1, 2]: we set \( d = 100 \) for P1 and P2. Finally, we set \( T_{p} = 10 \) for all test cases, following [12]. Due to the stochastic nature of the search algorithm, the QEA approach is run 20 times for each test case. All the experiments have been carried out in MATLAB software package, on a PC with Intel Core i5 of 3.4 GHz and 4 GB RAM.

**B. Results and comparisons to published results**

For each test case, the best, average and worst minimal cost values of the 20 experiment runs are recorded. Table 2 summarizes these results.

<table>
<thead>
<tr>
<th>Problem</th>
<th>( A_{0} )</th>
<th>Best</th>
<th>Average</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0.990</td>
<td>8.180</td>
<td>8.355</td>
<td>8.555</td>
</tr>
<tr>
<td></td>
<td>0.960</td>
<td>7.009</td>
<td>7.381</td>
<td>7.803</td>
</tr>
<tr>
<td>P2</td>
<td>0.990</td>
<td>5.423</td>
<td>5.901</td>
<td>6.477</td>
</tr>
<tr>
<td></td>
<td>0.960</td>
<td>15.870</td>
<td>15.923</td>
<td>16.087</td>
</tr>
<tr>
<td></td>
<td>0.975</td>
<td>14.770</td>
<td>14.893</td>
<td>15.237</td>
</tr>
<tr>
<td></td>
<td>0.995</td>
<td>12.855</td>
<td>12.999</td>
<td>13.126</td>
</tr>
</tbody>
</table>

Table 3 presents the detailed information about the best solution of each test case found by QEA approach out of 20 runs. The solution is represented in the form, ‘\( f(x_{i}) \)’ for a subsystem \( i \). For example, \( 3(2) \) in the top cell of the solution column ‘2’ indicates three type 3 components in subsystem 2. The reliability and cost values of each solution are presented as well.

<table>
<thead>
<tr>
<th>Problem</th>
<th>( A_{0} )</th>
<th>( A )</th>
<th>( C ($) )</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0.990</td>
<td>0.992</td>
<td>8.180</td>
<td>1(3) 1(3) 1(3) 4(2)</td>
</tr>
<tr>
<td></td>
<td>0.960</td>
<td>0.963</td>
<td>7.009</td>
<td>1(3) 2(1) 3(2) 3(1) 5(1)</td>
</tr>
<tr>
<td>P2</td>
<td>0.990</td>
<td>0.992</td>
<td>15.870</td>
<td>4(2) 6(1) 3(2) 3(2) 3(1) 7(3) 4(3)</td>
</tr>
<tr>
<td></td>
<td>0.980</td>
<td>0.980</td>
<td>14.770</td>
<td>4(2) 6(1) 3(2) 3(2) 3(1) 7(3) 3(2) 4(1)</td>
</tr>
<tr>
<td></td>
<td>0.975</td>
<td>0.976</td>
<td>12.855</td>
<td>4(2) 6(1) 5(6) 4(1) 7(3) 4(3)</td>
</tr>
</tbody>
</table>

In Table 4, the outcomes of QEA experiments are compared with published results, in terms of best solution quality and
number of fitness evaluations. Since various algorithms were tested using different computing facilities including the hardware platforms and software packages, the number of fitness evaluations is a more reliable metric of computational efficiency compared to the actual computation time. It is seen from Table 4 that across all test cases, the proposed approach requires the lowest numbers of fitness evaluations among the methods who have achieved the best solutions (i.e. GA, SP/TG, and PSP/LS). For P2 the proposed method is at least 10 times faster than other methods. As to P1, the proposed method is about twice faster than SP/TG and about 10 times faster than PSP/LS.

Note that we use the maximum number of fitness evaluations for SP/TG, since the exact number of fitness evaluations are not shown in the paper [9]. Due to the tabu search strategies, SP/TG might need less fitness evaluations in real applications. Nevertheless, the same type of Tabu search can be incorporated into our algorithm for further reduction of computation efforts. As to ACO, it has the smallest number of evaluations for P2, but it does not obtain the best solutions.

Table 4. Comparisons to the Published Results

<table>
<thead>
<tr>
<th>Publication</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td># of FE*</td>
<td>1.03e6</td>
<td>1.03e6</td>
</tr>
<tr>
<td>GA [7]</td>
<td>$\zeta$ (S)</td>
<td>15.870</td>
</tr>
<tr>
<td># of FE*</td>
<td>4.5e3</td>
<td>4.5e3</td>
</tr>
<tr>
<td>SP/TG [9]</td>
<td>$\zeta$ (S)</td>
<td>15.870</td>
</tr>
<tr>
<td># of FE &lt;2.05e4</td>
<td>2.05e4</td>
<td>2.05e4</td>
</tr>
<tr>
<td>PSP/LS [12]</td>
<td>$\zeta$ (S)</td>
<td>15.870</td>
</tr>
<tr>
<td># of FE &gt;1.0e5</td>
<td>1.0e5</td>
<td>1.0e5</td>
</tr>
<tr>
<td>QEA [12]</td>
<td>$\zeta$ (S)</td>
<td>15.870</td>
</tr>
<tr>
<td># of FE 1.1e4</td>
<td>1.1e4</td>
<td>1.1e4</td>
</tr>
</tbody>
</table>

* FE stands for fitness evaluation

V. CONCLUSIONS AND FUTURE WORKS

In this work, we have considered the SPMSS heterogeneous RAP. QEA is first introduced as the solution method. An efficient LS strategy is originally designed to enhance the exploitation ability of QEA. The validations on 6 benchmark test cases with comparisons to published results show that the proposed QEA approach is able to achieve the best solutions using much less computation resources than other methods. Given the promising results obtained, future works can be devoted to extending the application of this method to larger size SPMSS heterogeneous RAP or more complex RAPs, e.g. networks structure optimization, RAP under random-fuzzy environments, etc.

REFERENCES