Tuned Genetic Algorithms for Finding $p$-Medians of a Weighted Graph

A. Kaveh, M. Shahrouzi and Y. Naserifar

Abstract. There are various engineering applications dealing with the prototype problem of finding the best $p$-medians in a weighted graph. However, the heuristic developments are still of concern due to their complexity. This paper utilizes genetic algorithm as a well-known reliable evolutionary search for such a purpose. Problem formulation is studied, introducing a characteristic graph and specialized genotype representation called "Direct Index Coding". The genetic operators are also modified due to problem requirements, and further tuned using a simulated annealing approach. Such an enhanced evolutionary search tool is then applied to a number of examples to show its effectiveness regarding the exact results, and to compare efficiency between tuned and non-tuned GA.

Keywords: Genetic algorithm; $p$-median problem; Direct index coding; Simulated annealing; Parameter tuning.

INTRODUCTION

Many real world problems deal with optimal locating of a predetermined number of client nodes to serve other customer nodes in a given network. In other words, the common location finding of facilities such as emergency services, stores and shopping city centers, airports, universities and educational centers are all examples of such a layout optimization called P-Median Problem (PMP) [1]. It has already been further extended to some other engineering applications, such as domain decomposition, mesh generation and parallel computing [2-4].

The location finding models fall in an interesting field of operational research. These can be categorized in many ways, e.g. mini-max (center finding) or mini-sum problems. The latter is the case in this research; to locate $p$ server nodes among $N$ nodes of a given network, and allocate other customer nodes to them. Here, we consider the case where medians are associated with nodes of a network graph.

PMP is proven to be an NP-hard problem [5,6], that is no algorithm is known to practically solve its variants in polynomial time, as the size of the problem increases. Several approximate solutions are already employed in literature [7]. Consequently, heuristic and meta-heuristic search methods are of concern to many researchers [8-10]. Meta-heuristics usually can quickly reveal a near optimal solution without proof of its optimality. A survey of applied meta-heuristic methods for PMP is available in literature [11]. Novel utilizations of meta-heuristics, such as ACO or GA have also been investigated in engineering applications [2,12].

The most important issue in the meta-heuristic search is the balance between exploration and exploitation. This can be achieved using parameter tuning techniques, either via adaptive methods or by extensive hyper-optimization [13,14].

The genetic algorithm is a well-known meta-heuristic in which the explorative and exploitative agents are distinctly distinguished [15,16]. Such a feature has made it ideal for parameter tuning. In this article, an integer coded GA is tuned using a simulated annealing approach [17,18].

The outline of the paper is as follows. First, the problem formulation is described and a brief review of the exact method is given. Genetic operators are also
utilized for the PMP. The method is then applied to a number of examples and the results are compared with those of exact and pure GA, discussing efficiency and effectiveness.

**PROBLEM FORMULATION AND ITS CHARACTERISTIC GRAPH**

A graph $G$, denoted by $G = (N, E)$, consists of a non-empty set, $N$, of elements called nodes (vertices) and a set, $E$, of elements called edges (arcs), together with a relation of incidence, which associates each member with a pair of nodes called its “ends”. In a simple undirected graph, only one edge can be defined between a pair of nodes without considering priority, choice of which node is the first end of that edge. In some cases, some weight values are associated to the graph nodes, known as a “weighted graph”.

Consider a weighted undirected graph, $G = (N, E)$. The $p$-median problem is to find a subset, $N_p = \{i_1, \ldots, i_p\}$, of the total set of $N$ nodes, so that the following cumulative cost is minimized:

Minimize:

$$
\sigma_0 = \sum_{j \in N} \sum_{i \in N_p} v_j d(i, j),
$$

where $d(i, j)$ denotes the shortest distance from node $i$ to node $j$, and $v_j$ shows the weight of node $i$. Such a distance from each node to itself is taken as zero. For an undirected graph (in which $d(i, j) = d(j, i)$), the transmission of node $i$ is defined as $\sigma_0(i)$:

$$
\sigma_0(i) = \sum_{j \in N} v_j d(i, j).
$$

Number $\sigma_0(i)$ is a summation of the entries of row $i$ of the distance matrix obtained by multiplying every column $j$ of the distance matrix $[d(i, j)]$ by $v_j$. Node $i_0 \in N$ for which $\sigma_0(i_0)$ becomes minimum is the median of the undirected graph. Now, let $N_p$ be a subset of $N$ with $p$ nodes, then, each $N_p$ will have its own transmissions, similar to Equation 2, as follows:

$$
\sigma_0(N_p) = \sum_{j \in N} v_j d(N_p, j),
$$

in which:

$$
d(N_p, j) = \min_{i'} d(i', j).
$$

Let $i'$ be the node of $N_p$ that produces the minimum in Equation 3 then we say that node $j$ is allocated to $i'$. The optimization task is to find the subset with $p$ elements, $N_{p0} \subseteq N$, such that $\sigma_0(N_{p0})$ becomes minimum. Then, $N_{p0}$ will be a set containing the $p$-medians.

Let allocation matrix $[\xi_{ij}]$ be defined with each entry, $\xi_{ij}$, of 1, if node $j$ is allocated to median node $i$, and 0 otherwise. The alternate integer programming formulation for the PMP will be:

Minimize:

$$
\sigma = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \xi_{ij},
$$

Subject to:

$$
\sum_{i=1}^{n} \xi_{ij} = 1 \quad \text{for } j = 1, 2, \ldots, n, \tag{6}
$$

$$
\sum_{i=1}^{n} \xi_{ii} = p, \tag{7}
$$

$$
\xi_{ij} \leq \xi_{ii} \quad \text{for all } i, j = 1, 2, \ldots, n, \tag{8}
$$

$$
\xi_{ij} = 0 \quad \text{or} \quad 1. \tag{9}
$$

In the above relationships, $[W_{ij}]$ is assumed to be the weighted distance matrix, i.e., it is the distance matrix with every column $j$ multiplied by $v_j$. Equation 6 ensures that any given node $j$ is allocated to one and only one median $i$. Equation 7 ensures that there are only $p$ medians, and Equation 8 guarantees that allocations are made only to the median nodes.

A $p$-median problem framework can be represented using a string with $p$ cells, when each cell is to be filled with non-replicate node numbers in the integer range $\{1, 2, \ldots, N\}$. Alternatively, such a framework addresses a bi-partite characteristic graph with $p$ vertices in its 1st part, which are to be connected with $p$ out of $N$ vertices in the second part (Figure 2b). This way, any $p$-edge matching of this characteristic graph represents the corresponding evaluated string as a solution candidate of the problem (Figure 2).

Any movement of the graph nodes can neither disturb its adjacency nor transform it to a new graph. Similarly, any substitution in arrangement of the filled cells or any permutation of a set of $p$ nodes will not generate a new solution. Thus, the order of the node numbers in such a $p$-cell string has no importance for the PMP, and permutation in it is allowed.

**EXACT ALGORITHM FOR PMP**

Cardinality of the PMP search space, i.e., $\binom{N}{p}$, depends on factorials of $N$ and $p$. Consequently, searching all these possible combinations will be an extensive time-consuming or even impractical task for
larger problem sizes. It is proven that finding the medians of a planar graph is an NP-hard problem [5,6].
For the sake of simplicity, the Floyd-Warshall algorithm, given via the following steps [19], is given here. However, more efficient methods may also be found in literature [20-22]:

1. Construct all alternative combinations of \( \binom{N}{p} \) as candidate median lists.
2. Allocate other graph nodes to their corresponding medians in every such median list.
3. Compute cost of every median list.
4. Among all possible median lists, identify the one having the least cost as the solution of the PMP and report allocated nodes to these optimum p-median.

**GENETIC SEARCH UTILIZATION FOR PMP**

As a well-proven evolutionary search, a Genetic Algorithm (GA) was systematically introduced and studied by Holland in 1975 [15]. GA works in the search space of coded variables called “genotypes” instead of corresponding physical ones, i.e. phenotypes. Such a feature has made it a general search tool for various problem fields, as it enables genotypic jumping over different local search islands seeking the global optimum. Hence, the encoding scheme plays a crucial role in the genetic search.

Binary coding was the first to be employed in the standard GA due to its simplicity and minimal choice of allele variation for every gene, that is only 2 options (say 0 and 1). However, it is not the most suitable coding for many real-world phenomena which are not of binary type. In such cases including PMP, binary representation can disaffect optimization in a number of ways:

- Resulting in larger genotypic search space than needed.
- Hamming cliffs effect.
- Hidden mutation in binary crossover.
- Uncontrolled mutation range in the decoded phenotype that randomly varies with the location of a mutated bit in the corresponding genotype.

Detailed reasoning for such effects is already available in literature [23]. As a solution, Direct Index Coding (DIC) is utilized to suit for cases when the range of allele variation is a list of property indices as the alphabet for the allele assignment [23]. Since the PMP falls in such a category, DIC has been used for the current problem.

According to this type of coding, the chromosome will be identical to the pre-mentioned string with p-genes. An evaluated genotype will be a p-gene chromosome in which any gene-value (allele) is a node number chosen from a list of integer indices between 1 and N (Figure 2).

The number of medians in PMP is fixed to p, hence no duplicate alleles in a chromosome are permitted. Thus, the genetic operators should be modified in order to efficiently satisfy the PMP requirements. The specialized GA operators are developed in the current research as follows.
Initiation

In order to get most benefit from GA exploration capabilities, it is desired to start the 1st population with randomly initiated individuals with no occurrence of replicate alleles in each chromosome. Suppose the number of chromosomes in every generation is set to PopSize. The above requirements can be simply satisfied, generating PopSize number of permuted strings of indices between 1 and N. Then, the first p indices of each string are selected as the corresponding chromosome of the 1st generation.

Crossover

The role of crossover is to search all available exchanges between genes of the population, that is exploiting the corresponding local search island. Although each chromosome of the previous generation is a permitted genotype, the simple crossover between 2 parents may generate ill-chromosomes with duplicate alleles (Figure 3). In order to avoid such a malfunction, as soon as every 2 chromosomes are selected as candidate parents, one of them is permuted so that any pair of their genes with the same allele takes the same place order. This way, no duplicate alleles will arise in the resulted children (Figure 4).

Mutation

Crossover by itself only exploits a subset of the whole-search space related to the property of the current population. Another operator will be then necessary to explore other search regions toward the one containing the global optimum. The allele of any gene in a chromosome should have random chance of being exchanged with other permitted values in the range with the predetermined probability threshold of $P_m$. If the candidate list for allele mutation is taken as $\{1, 2, \ldots, N\}$, duplicate values may be entered to the same chromosome (Figure 5). Once a mutation is ordered, the corresponding allele range is thus modified by omitting other current alleles of that chromosome from it. This way, allele duplication is completely avoided during mutation (Figure 6).

SIMULATED ANNEALING APPROACH FOR TUNING THE SEARCH

Another important issue in implementation of a genetic algorithm is parameter tuning to provide proper balance between exploitation and exploration. Lack of such a balance may lead either to inefficiency or even to premature convergence toward local optima. In the earlier generations, it is desired to diversify in order to capture more representatives of the whole search space regions. This can be achieved using a lower ratio of exploitative to explorative operators, i.e. a crossover threshold with respect to mutation. In the last iterations, however, less mutation probability is required to allow proper search intensification.

A number of attempts have already been reported regarding this issue including extensive multi-parameter hyper optimization [17,18]. Here, a Simulated Annealing (SA) approach is employed as a single-stage modifier. In SA methodology, the probability threshold of diversification is gradually decreased according to the following relation, as an artificial temperature, $T$, increases with iteration counter, $k$.
\[ f_c(k) = e^{-\frac{\Delta E}{k T}} \]  

(10)

in which the term \( \Delta E \) denotes changes in objective function due to perturbing the design variable vector, and the artificial Boltzmann constant, \( C_B \), is a control parameter. \( k \) stands as counter to the search iterations.

Using the same approach, \( f_e \) is taken as an envelope function which modifies the desired parameter(s) of the current genetic search and is determined in any \( k \)th generation as:

\[ f_e(k) = \begin{cases} 1, & \text{for } k = 1 \\ e^{-\frac{\Delta E}{k T}}, & \text{whereas: } T = \frac{N}{\sum_{i=1}^{\infty} -1.0, \text{ for: } k>1} \end{cases} \]  

(11)

The envelope function starts at unity, and the Boltzmann coefficient, \( C_B \), is decided for each case.

Such an envelope function is used to gradually decrease \( P_m \) and increase \( P_c \) as depicted in Figure 7 for a sample Boltzmann constant. \( P_m \) stands for mutation probability and \( P_c \) is the crossover probability. This way, the GA will diversify in the earlier generations to capture more representatives of the whole search space, while suitable search intensification is adjusted, as the optimization gets closer to its end.

**NUMERICAL EXAMPLES**

In this section, some numerical examples are studied to show the effectiveness and efficiency of the proposed method. The elitism strategy is also employed to save the best individual of every previous generation and replace it with the worst one of the current population. Tournament selection is used, not to omit the chance of less fit, but probably to select good parents during reproduction. Modified one-point crossover and mutation are utilized for the current PMP on undirected graphs.

The control parameters for the pure GA are
selected after a number of trials and given in Table 1. For the proposed Hybrid GA and SA, the Boltzmann coefficient is taken as 0.25 and other controlling parameters are set as in Table 2.

For every case, the best and average results are reported out of a number of (10) consequent runs. The thresholds of crossover rate, \( P_c \), and mutation rate, \( P_m \), are tuned using the SA approach as mentioned in Figure 7.

Four different \( p \) values: 2, 5, 10 and 20 for constant \( N \) are treated as distinct \( p \)-median problems. GA results are also compared to the exact algorithm when converged in a reasonable time. For the sake of conciseness, only more important results are briefed in Tables 3 to 6.

**Example 1-1**

It is a classic sample for PMP with 8 nodes [14,16]. The graph and its nodal/edges’ weights are depicted in Figure 8. Assume that the median list \{1,3,7\} be a solution for 3-median finding in this problem. Figure 9a shows both the corresponding chromosome and its characteristic graph. Assume that according to Figure 9b, nodes \{2,4\} are associated with node 1,

---

**Table 1. Control parameters of the GA.**

<table>
<thead>
<tr>
<th>Method</th>
<th>PopSize</th>
<th>( P_m )</th>
<th>( P_c )</th>
<th>Number of Generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td>30</td>
<td>0.1</td>
<td>0.9</td>
<td>100000</td>
</tr>
</tbody>
</table>

---

**Table 2. Control parameters for the Hybrid GA and SA.**

<table>
<thead>
<tr>
<th>Method</th>
<th>PopSize</th>
<th>Boltzman Coef. ( C_b )</th>
<th>Initial ( P_m )</th>
<th>Initial ( P_c )</th>
<th>Target Limit on ( P_m )</th>
<th>Target Limit on ( P_c )</th>
<th>Number of Generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>30</td>
<td>0.25</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0.9</td>
<td>100000</td>
</tr>
</tbody>
</table>
Table 3. Algorithms results for Example 2 with 45 nodes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA + SA</td>
<td>21 26</td>
<td>104</td>
<td>0.03</td>
<td>104</td>
<td>453</td>
<td>0.05</td>
</tr>
<tr>
<td>GA</td>
<td>21 26</td>
<td>104</td>
<td>0.02</td>
<td>104</td>
<td>444</td>
<td>0.05</td>
</tr>
<tr>
<td>Exact</td>
<td>20 25</td>
<td>104</td>
<td>0.06</td>
<td>104</td>
<td>990</td>
<td>0.06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA + SA</td>
<td>11 14 17 30 34</td>
<td>63</td>
<td>0.08</td>
<td>63</td>
<td>16479</td>
<td>0.75</td>
</tr>
<tr>
<td>GA</td>
<td>11 14 17 34 30</td>
<td>63</td>
<td>0.12</td>
<td>63</td>
<td>7845</td>
<td>0.70</td>
</tr>
<tr>
<td>Exact</td>
<td>3 16 29 32 35</td>
<td>63</td>
<td>1.2300</td>
<td>63</td>
<td>1221759</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA + SA</td>
<td>2 6 13 17 19 24 30 36 41 43</td>
<td>38</td>
<td>0.67</td>
<td>38</td>
<td>77808</td>
<td>6.12</td>
</tr>
<tr>
<td>GA</td>
<td>2 7 13 18 19 24 30 35 41 44</td>
<td>38</td>
<td>0.48</td>
<td>38</td>
<td>72756</td>
<td>8.26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA + SA</td>
<td>1 4 7 8 11 14 15 17 18 20 22 23 25 26 28 31 32 39 42 43</td>
<td>25</td>
<td>0.02</td>
<td>25</td>
<td>39</td>
<td>0.03</td>
</tr>
<tr>
<td>GA</td>
<td>2 4 5 8 9 11 16 18 19 21 22 23 31 32 35 38 39 43 44 45</td>
<td>25</td>
<td>0.02</td>
<td>25</td>
<td>33</td>
<td>0.04</td>
</tr>
</tbody>
</table>

while no node is associated with 7. The remainder is associated to node 3.

Example 1-2

It is the same as Example 1-1, but solved for 2-medians. For this example, the symmetric shortest distance matrix is given as below:

\[
\begin{bmatrix}
0 & 4 & 3 & 2 & 6 & 5 & 8 & 6 \\
4 & 0 & 2 & 3 & 4 & 4 & 7 & 5 \\
3 & 2 & 0 & 1 & 3 & 2 & 5 & 3 \\
2 & 3 & 1 & 0 & 4 & 3 & 6 & 4 \\
6 & 4 & 3 & 4 & 0 & 5 & 8 & 6 \\
5 & 4 & 2 & 3 & 5 & 0 & 3 & 1 \\
8 & 7 & 5 & 6 & 8 & 3 & 0 & 3 \\
6 & 5 & 3 & 4 & 6 & 1 & 3 & 0
\end{bmatrix}
\]
Table 4. Algorithms results for Example 3 with 95 nodes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>23 70</td>
<td>334</td>
<td>0.03</td>
<td>334</td>
<td>2409</td>
<td>0.21</td>
</tr>
<tr>
<td>GA</td>
<td>23 70</td>
<td>334</td>
<td>0.05</td>
<td>334</td>
<td>1815</td>
<td>0.15</td>
</tr>
<tr>
<td>Exact</td>
<td>23 70</td>
<td>334</td>
<td>0.33</td>
<td>334</td>
<td>4465</td>
<td>0.22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>8 27 44 73 78</td>
<td>201</td>
<td>0.27</td>
<td>201.2</td>
<td>67533</td>
<td>4.60</td>
</tr>
<tr>
<td>GA</td>
<td>8 27 44 73 78</td>
<td>201</td>
<td>0.23</td>
<td>201.2</td>
<td>44997</td>
<td>3.34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>9 12 29 32 47 50 64 77 80 83</td>
<td>134</td>
<td>15.94</td>
<td>135.4</td>
<td>162525</td>
<td>16.15</td>
</tr>
<tr>
<td>GA</td>
<td>7 14 36 29 47 50 64 77 80 83</td>
<td>134</td>
<td>2.46</td>
<td>134.3</td>
<td>241956</td>
<td>31.78</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>4 7 15 16 18 25 27 34 36 43 50 53 57 65 69 72 76 84 88 91</td>
<td>82</td>
<td>82.81</td>
<td>83.1</td>
<td>244893</td>
<td>51.11</td>
</tr>
<tr>
<td>GA</td>
<td>4 7 15 16 18 25 27 34 36 43 50 53 57 62 69 75 76 81 88 94</td>
<td>82</td>
<td>96.03</td>
<td>83</td>
<td>384018</td>
<td>122.81</td>
</tr>
</tbody>
</table>

Nodal cost vector, $V$, is as follows:

$$V = \begin{bmatrix} 3 & 1 & 2 & 3 & 1 & 0 & 4 & 1 \end{bmatrix},$$

and the weighted distance matrix, $W$, will be:

$$W = \begin{bmatrix} 0 & 4 & 6 & 6 & 6 & 0 & 32 & 6 \\ 12 & 0 & 4 & 9 & 4 & 0 & 28 & 5 \\ 9 & 2 & 0 & 3 & 3 & 0 & 20 & 3 \\ 6 & 3 & 2 & 0 & 4 & 0 & 24 & 4 \\ 18 & 4 & 6 & 12 & 0 & 0 & 32 & 6 \\ 15 & 4 & 4 & 9 & 5 & 0 & 3 & 1 \\ 24 & 7 & 10 & 18 & 8 & 0 & 3 & 3 \\ 18 & 5 & 6 & 12 & 6 & 0 & 12 & 0 \end{bmatrix}.$$  

The whole alternatives for 2-medians are listed as:

(1 2), (1 3), (1 4), (1 5), (1 6), (1 7), (1 8),

(2 3), (2 4), (2 5), (2 6), (2 7), (2 8).

With corresponding costs of:

Cost = $[47, 31, 37, 54, 32, 25, 34, 38, 40, 58, 42, 32, 41, 34, 37, 30, 29, 39, 28, 18, 27, 45, 43, 52, 38, 49, 47].$

The solution will be (4, 7) for $p = 2$, leading to the least cost. Due to the tiny size of such an illustrative problem, the same result could be achieved by all of
Table 5. Algorithms results for Example 4 with 190 nodes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>90 101</td>
<td>1040</td>
<td>0.17</td>
<td>1041.2</td>
<td>27429</td>
<td>1.99</td>
</tr>
<tr>
<td>GA</td>
<td>90 101</td>
<td>1040</td>
<td>0.05</td>
<td>1040.2</td>
<td>266838</td>
<td>16.55</td>
</tr>
<tr>
<td>Exact</td>
<td>90 101</td>
<td>1040</td>
<td>0.66</td>
<td>1040</td>
<td>17955</td>
<td>0.80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>31 43 78 135 165</td>
<td>600</td>
<td>0.56</td>
<td>601.3</td>
<td>49293</td>
<td>5.42</td>
</tr>
<tr>
<td>GA</td>
<td>26 56 113 128 160</td>
<td>600</td>
<td>0.34</td>
<td>600.9</td>
<td>12531</td>
<td>1.41</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>25.33 49 62 78 113 140 148 156 164</td>
<td>404</td>
<td>3.09</td>
<td>401.3</td>
<td>125355</td>
<td>23.69</td>
</tr>
<tr>
<td>GA</td>
<td>25.33 49 58 62 113 129 142 158 166</td>
<td>404</td>
<td>6.54</td>
<td>401.3</td>
<td>282099</td>
<td>51.26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>22 25 29 33 36 39 47 51 77 83 92 105 115 126 139 142 152 169 176 185</td>
<td>208</td>
<td>246.53</td>
<td>269.7</td>
<td>530982</td>
<td>165.41</td>
</tr>
<tr>
<td>GA</td>
<td>6 16 22 31 49 53 59 65 77 82 104 114 126 138 142 152 155 160 169 185</td>
<td>208</td>
<td>592.24</td>
<td>270.9</td>
<td>961323</td>
<td>306.16</td>
</tr>
</tbody>
</table>

The exact, GA and Hybrid GA methods in fractions of a second.

\[
\xi = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}
\]

Example 2

In this example, a 45 node graph is treated with the algorithms (Figure 10). The nodal weights are taken as unity and the weighted distance matrix is computed, according to [12].

Results for different \( p \) values of \{2, 5, 10 and 20\} are obtained and summarized in Table 3. For \( p = 2 \), the GA and Hybrid GA both converged to the median list with the same optimum cost of the exact solution. The matter shows the effectiveness of the proposed meta-heuristic search. Besides, the difference in required computational time has been highly increased for \( p = 5 \). For larger problem sizes, results of the proposed exact methods were not captured in the same practical time, due to the NP-hard nature of the problem.
Table 6. Algorithms results for Example 5 with 760 nodes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>183 583</td>
<td>6354</td>
<td>0.61</td>
<td>6354</td>
<td>8745</td>
<td>1.88</td>
</tr>
<tr>
<td>GA</td>
<td>178 368</td>
<td>6354</td>
<td>0.08</td>
<td>6354</td>
<td>9165</td>
<td>1.58</td>
</tr>
<tr>
<td>Exact</td>
<td>178 368</td>
<td>6354</td>
<td>37.05</td>
<td>6354</td>
<td>288420</td>
<td>34.71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>65 277 445 485 585</td>
<td>3774</td>
<td>10.73</td>
<td>3774</td>
<td>124353</td>
<td>40.58</td>
</tr>
<tr>
<td>GA</td>
<td>104 266 386 646 656</td>
<td>3774</td>
<td>12.23</td>
<td>3774</td>
<td>348330</td>
<td>115.69</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>55 88 266 278</td>
<td>2479</td>
<td>199.40</td>
<td>2494.3</td>
<td>248802</td>
<td>149.88</td>
</tr>
<tr>
<td>GA</td>
<td>55 88 266 278</td>
<td>2479</td>
<td>263.28</td>
<td>2488.6</td>
<td>468939</td>
<td>294.80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Median List</th>
<th>Best Cost</th>
<th>Best Time (s)</th>
<th>Average Cost</th>
<th>Average Number of Fitness Evaluations</th>
<th>Average Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA+SA</td>
<td>50 63 77 174</td>
<td>1632</td>
<td>703.91</td>
<td>1630.5</td>
<td>540972</td>
<td>608.92</td>
</tr>
<tr>
<td>GA</td>
<td>29 56 63 149</td>
<td>1637</td>
<td>1640</td>
<td>1646</td>
<td>1394229</td>
<td>1696.89</td>
</tr>
</tbody>
</table>

Example 3

Figure 11 shows the 95 node graph as the next example [12]. Again, the nodal weights are taken as unity. Results for different p values are summarized in Table 4. For the small p of 2, all the tree methods converged to the same exact solution. However, for larger p values, the superiority of Hybrid GA over GA is further declared in the resulted computational time.

Example 4

The 190 node graph of this example [12] is shown in Figure 12. Table 5 summarizes the best and average obtained results for different numbers of medians. In this example, the Hybrid GA could improve not only the computational time, but also the average cost with respect to pure GA.
Example 5

This 760 node example is also treated to study the capability of algorithms working on larger size problems [12]. As shown in Figure 13, the corresponding graph has 4 inner holes. Again, the nodal weights are taken as unity. Results for different p values are summarized in Table 6.

DISCUSSION AND CONCLUSION

As can be realized from the results, when the number of nodes and medians decrease, the probability of capturing the global optimum increases. For larger size problems, the result of Hybrid GA is generally better than GA.

Figures 14 and 15 show a smoother and more stable convergence of Hybrid GA than pure GA. Figure 14 also illustrates the probability of premature convergence for the pure GA, despite the proposed Hybrid GA.

According to the results in Figures 16 and 17, the Hybrid GA is more efficient than GA for various p values. Comparison of the best CPU time for different examples in Figure 18 confirms the superiority of the developed Hybrid GA over the pure GA. It also shows that the pure GA is more sensitive to the number of nodes as a factor of search space cardinality. The conclusion stays reliable when considering average required time, even with smoother curves in Figure 19.

Generally, it is noted that the proposed exact solution for larger size NP-hard PMP’s may not be
Figure 16. Comparison of algorithms efficiency of best achieved results for various number of medians in 760-node example.

Figure 17. Comparison of algorithms efficiency in average number of fitness evaluations or various number of medians in 760-node example.

Figure 18. Comparison of algorithms efficiency for best achieved results and various number of nodes \( N = \{45, 95, 190, 760\} \).

achievable via practical time. Use of a meta-heuristic and stochastic search is of concern not only for quicker convergence, but also for jumping over the local optima, in order to provide search effectiveness and a better quality of results. However, the application of standard GA is not recommended due to its higher sensitivity to the problem size. Instead, the developed Hybrid GA can provide a proper balance between effectiveness and efficiency of such a combinatorial optimization task due to the proposed tuning of exploitative and explorative operators.

ACKNOWLEDGMENT
The first author is grateful to Iran National Science Foundation for support.

REFERENCES

**BIOGRAPHIES**

**Ali Kaveh** was born in 1948 in Tabriz, Iran. After graduation from Department of Civil Engineering at the University of Tabriz in 1969, he continued his studies on Structures at Imperial College of Science and Technology, London University, and received his M.S., DIC and Ph.D. in 1970 and 1974, respectively. He then joined the Iran University of Science and Technology in Tehran, and presently he is Professor of Structural Engineering at this university. Professor Kaveh is the author of 200 papers published in international journals and 125 papers presented at international conferences. He has had 23 books in Farsi and 3 books in English; published by Wiley, American Mechanical Society and Research Studies Press.

**Mohsen Shahrourzi** was born in 1975 in Tehran, Iran. He completed his B.S. in Civil Engineering and his M.S. in Earthquake Engineering at Sharif University of Technology in 1997 and 2000, respectively. He continued his studies and received his Ph.D. from the International Institute of Earthquake Engineering and Seismology in 2006. He then, began his cooperation with the Building and Housing Research Center in Tehran, and in 2008 joined the Tarbiat Moallem University in Tehran as Assistant Professor of Earthquake Engineering. Dr Shahrourzi is the author of 23 papers, 8 papers being published in international journals and 3 presented at international conferences.

**Yaser Naserifar** was born in 1981 in Khoramabad, Iran. He received his B.S. degree from the Department of Civil Engineering at the Azad University of Arak, in 2006. He, then, continued his studies at the Department of Earthquake Engineering at the Building and Housing Research Center, Tehran, receiving his M.S. in 2009. He has worked in the Structural Laboratory of the Building and Housing Research Centre for three years.