Transaction A: Civil Engineering
Vol. 17, No. 5, pp. 350–362

SCIENTIA IRANICA

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

A. Kaveh[1,*], M. Shahrouzi[2,3] and Y. Naserifar[3]

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

1. *Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Tehran, P.O. Box 16846-13114, Iran.*
2. *Department of Civil Engineering, Tarbiat Moallem University, Tehran, P.O. Box 15614, Iran.*
3. *Department of Engineering, Building and Housing Research Center, Tehran, P.O. Box 13145-1696, Iran.*
*. *Corresponding author. E-mail: alikaveh@iust.ac.ir*

Received 2 February 2010; received in revised form 20 May 2010; accepted 3 July 2010

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, $N_p = \{i_1, \cdots, i_p\}$, of the total set of N nodes, so that the following cumulative cost is minimized:

Minimize:

$$\sigma_0 = \sum_{j \in N} \sum_{i \in N_p} v_j d(i, j), \tag{1}$$

where $d(i, j)$ denotes the shortest distance from node i to node j, and v_i 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 $\sigma_0(i)$:

$$\sigma_0(i) = \sum_{j \in N} v_j d(i, j). \tag{2}$$

Number $\sigma_0(i)$ is a summation of the entries of row i of the matrix; obtained by multiplying every column j of the distance matrix $[d(i, j)]$ by v_j. Node $\bar{i}_0 \in N$ for which $\sigma_0(\bar{i}_0)$ 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:

$$\sigma_0(N_p) = \sum_{j \in N} v_j d(N_p, j), \tag{3}$$

in which:

$$d(N_p, j) = \min_{\substack{i' \in Np \\ j \in N}} d(i', j). \tag{4}$$

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, $\bar{N}_{p0} \in N$, such that $\sigma_o(\bar{N}_{po})$ becomes minimum. Then, $\bar{N}_{p0}$ will be a set containing the p-medians.

Let allocation matrix $[\xi_{ij}]$ be defined with each entry, ξ_{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:

$$\sigma = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \xi_{ij}. \tag{5}$$

Subject to:

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

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

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

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

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 $\{1, 2, \cdots, N\}$. 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 1b). 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

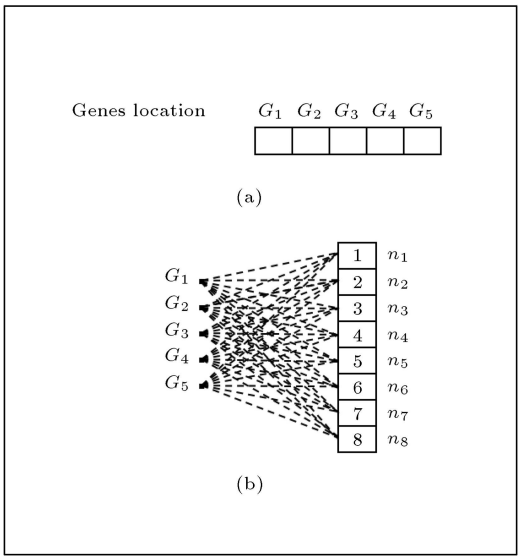


Figure 1. Representation framework of PMP with (a) a p-cell string/chromosome; and (b) a complete bipartite characteristic graph ($p = 4$).

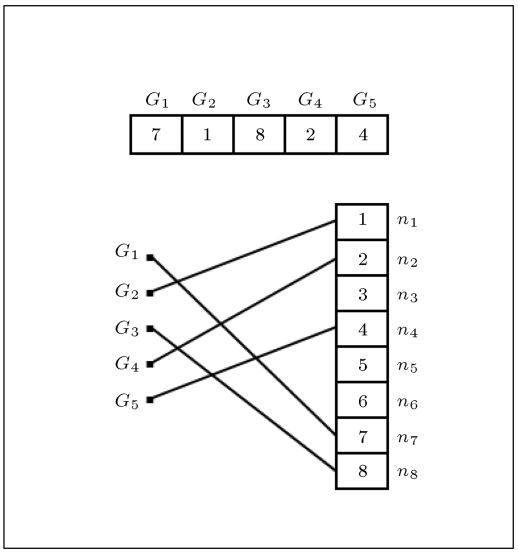


Figure 2. A candidate solution of a sample PMP; represented with an evaluated string and its corresponding matching graph.

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-medians.

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 permutated 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

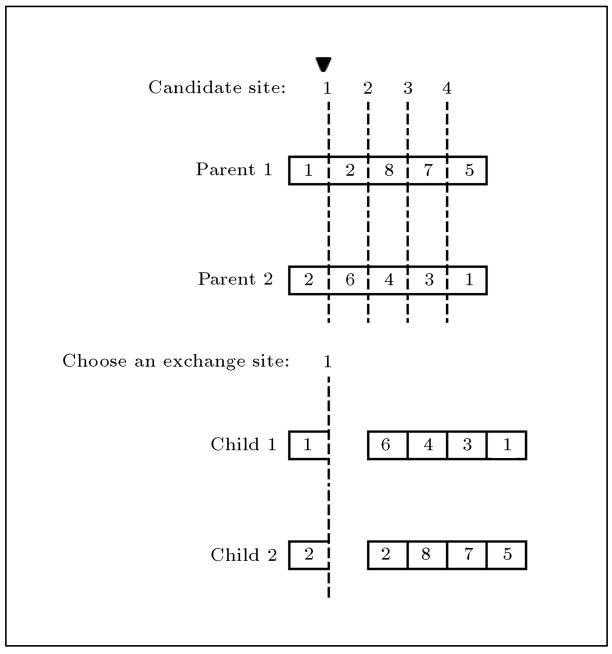


Figure 3. One-point crossover on sample pair of correct parent chromosomes can result in incorrect chromosomes; each one with duplicate gene values.

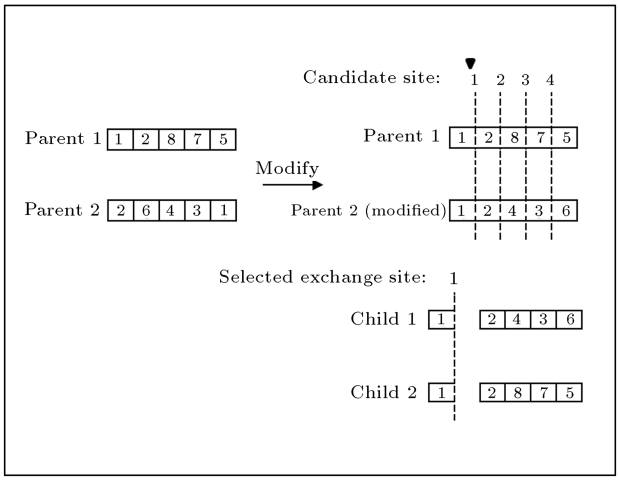


Figure 4. Modifying parent chromosomes via crossover has insured generating correct children.

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 $\{1, 2, \cdots, N\}$, 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:

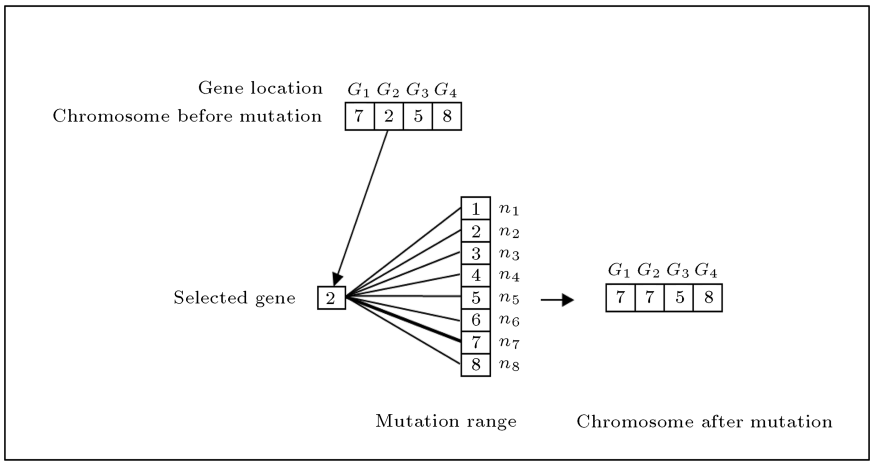


Figure 5. Example of simple mutation which has generated an incorrect chromosome with duplicate gene values.

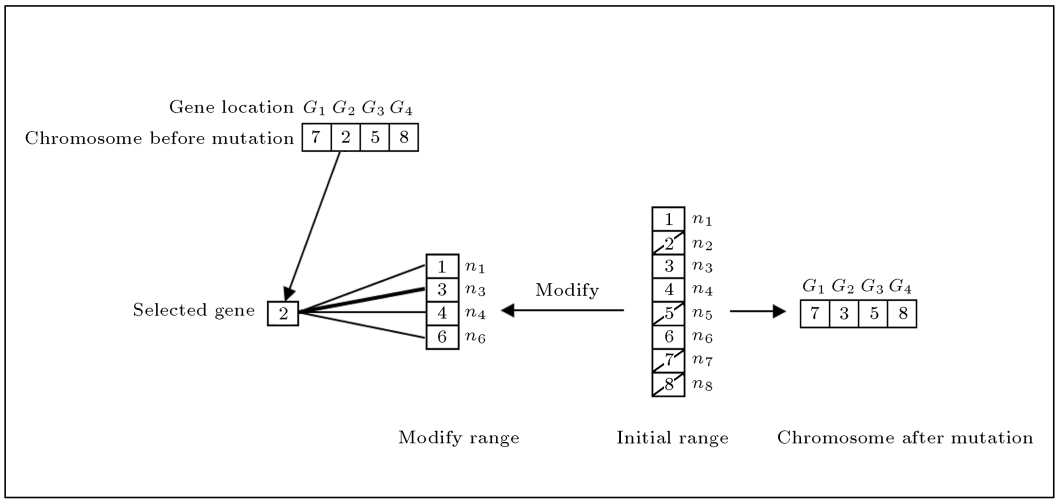


Figure 6. Modified mutation range has insured production of correct mutated chromosome.

$$f_e(k) = e^{\frac{-\triangle E}{C_B T(k))}}, \tag{10}$$

in which the term ΔE denotes changes in objective function due to perturbing the design variable vector, and the artificial Boltzman 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 kth generation as:

$$f_e(k) = \begin{cases} 1, & \text{for: } k = 1 \\ e^{\frac{-1}{C_B T}}, & \text{whereas: } T = \frac{N}{k-1} - 1.0, \text{ for: } k > 1 \end{cases} \tag{11}$$

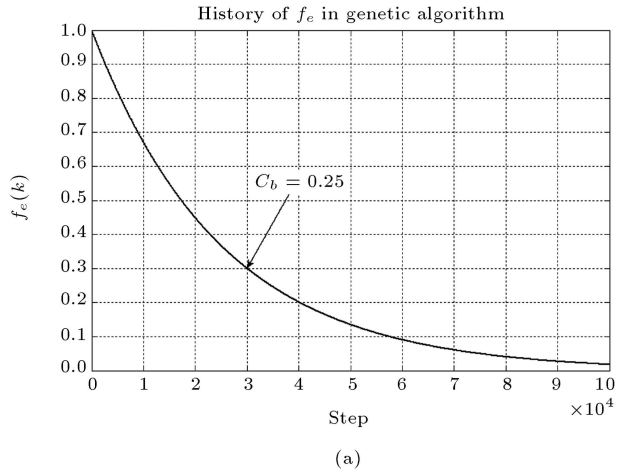
The envelope function starts at unity, and the Boltzman coefficient, C_B, is decided for each case.

Such an envelope function is used to gradually decrease P_m and increase P_c as depicted in Figure 7 for a sample Boltzman 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.

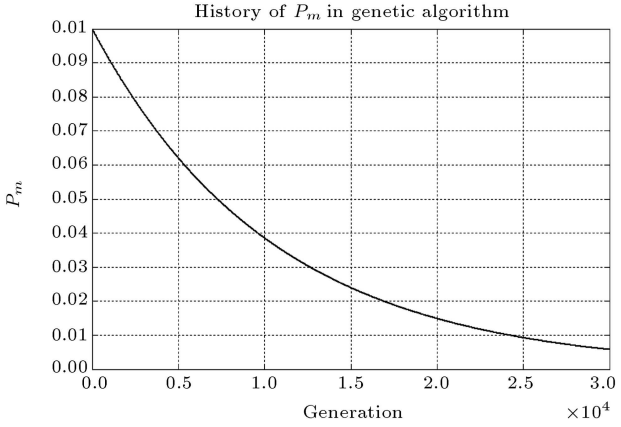
NUMERICAL EXAMPLES

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.

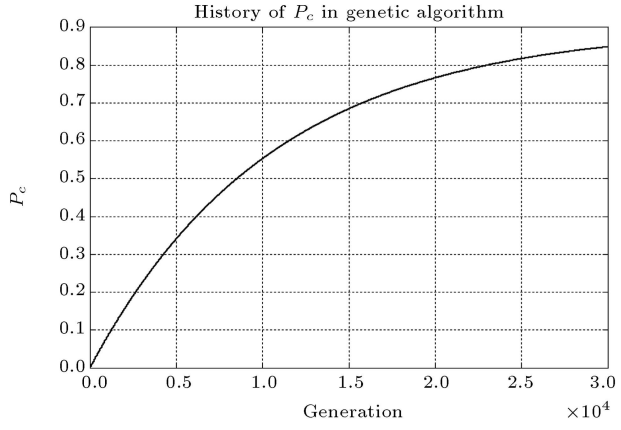
The control parameters for the pure GA are



(a)



(b)



(c)

Figure 7. Tuning curves for Hybrid GA. (a) The envelope function; (b) mutation probability; and c) crossover probability.

Table 1. Control parameters of the GA.

Method	PopSize	P_m	P_c	Number of Generations
GA	30	0.1	0.9	100000

selected after a number of trials and given in Table 1. For the proposed Hybrid GA and SA, the Boltzman 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,

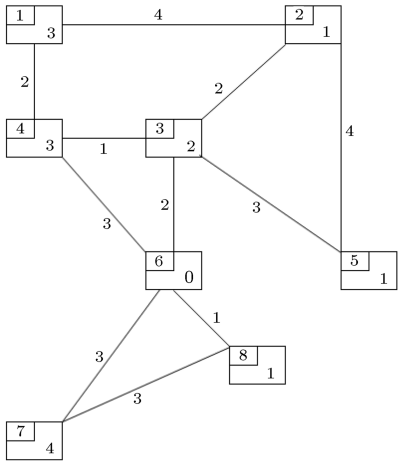


Figure 8. Example 1 with weights of its vertices and edges.

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

Method	PopSize	Boltzman Coeff. C_b	Initial P_m	Initial P_c	Target Limit on P_m	Target Limit on P_c	Number of Generations
GA+SA	30	0.25	0.1	0	0	0.9	100000

Table 3. Algorithms results for Example 2 with 45 nodes.

Method	Median List	Best Cost	Best Time (s)	Average Cost	Average Number of Fitness Evaluations	Average Time (s)
GA+SA	21 26	104	0.03	104	453	0.05
GA	21 26	104	0.02	104	444	0.05
Exact	20 25	104	0.06	104	990	0.06
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	11 14 17 30 34	63	0.08	63	16479	0.75
GA	11 14 17 34 39	63	0.12	63	7845	0.70
Exact	3 16 29 32 35	63	12300	63	1221759	50
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	2 6 13 17 19 24 30 36 41 43	38	0.67	38	77898	6.12
GA	2 7 13 18 19 24 30 35 41 44	38	0.48	38	72756	8.26
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	1 4 7 8 11 14 15 17 18 20 22 23 25 26 28 31 32 39 42 45	25	0.02	25	39	0.03
GA	2 4 5 8 9 11 16 18 19 21 22 23 31 32 35 38 39 43 44 45	25	0.02	25	33	0.04

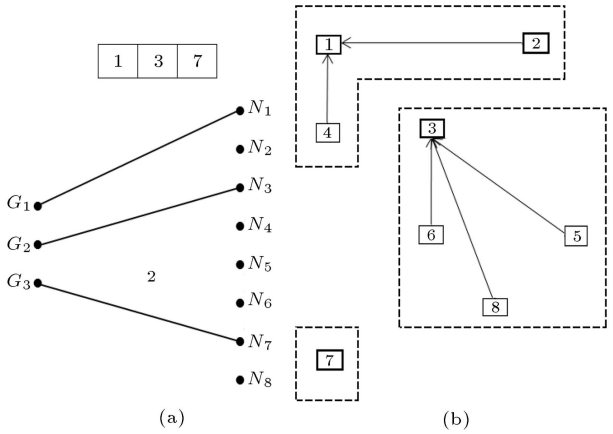


Figure 9. a) A chromosome as a sample 3-median list and its corresponding characteristic graph for Example 1. b) Sample node allocation to the median nodes.

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.

Method	Median List	Best Cost	Best Time (s)	Average Cost	Average Number of Fitness Evaluations	Average Time (s)
GA+SA	23 70	334	0.03	334	2409	0.21
GA	23 70	334	0.05	334	1815	0.15
Exact	23 70	334	0.33	334	4465	0.22
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	8 27 44 73 78	201	0.27	201.2	67533	4.60
GA	8 27 44 73 78	201	0.23	201.1	44937	3.34
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	9 12 29 32 47 50 64 77 80 83	134	15.94	135.4	162525	16.15
GA	7 14 26 29 47 50 64 77 80 83	134	2.46	134.3	241956	31.78
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	4 7 15 16 18 25 27 34 36 43 50 53 57 65 69 72 76 84 88 91	82	82.81	83.1	244893	51.11
GA	4 7 15 16 18 25 27 34 36 43 50 54 57 62 69 75 76 81 88 94	82	96.03	83	384018	122.81

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),

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

(4 5), (4 6), (**4 7**), (4 8),

(5 6), (5 7), (5 8),

(6 7), (6 8),

(7 8).

With corresponding costs of:

Cost = [47, 31, 37, 54, 32, 25, 34, 38, 40, 58, 42, 32, 41,

34, 37, 30, 20, 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.

Method	Median List	Best Cost	Best Time (s)	Average Cost	Average Number of Fitness Evaluations	Average Time (s)
GA+SA	90 101	1040	0.17	1041.2	27429	1.99
GA	90 101	1040	0.05	1040.2	266838	16.55
Exact	90 101	1040	0.66	1040	17955	0.80
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	31 43 78 135 165	600	0.56	601.3	49293	5.42
GA	26 56 113 128 160	600	0.34	600.9	12531	1.41
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	25 33 49 62 78 113 140 148 156 164	404	3.09	404.3	125355	23.69
GA	25 33 49 58 62 113 129 142 158 166	404	6.54	404.3	282939	54.26
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	22 25 29 33 36 39 47 51 77 83 92 105 115 126 139 142 152 169 176 185	268	246.53	269.7	530982	165.41
GA	6 16 22 31 49 53 59 65 77 82 104 114 126 138 142 152 155 160 169 185	268	592.24	270.9	961323	396.16

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

$$\xi = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

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

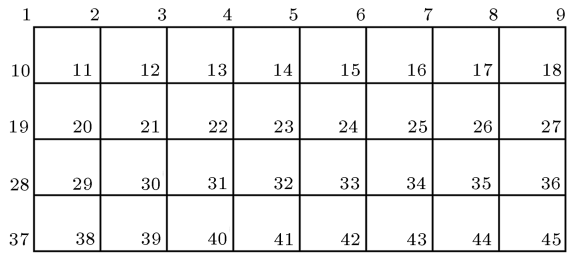


Figure 10. Graph model of Example 2 with 45 nodes.

$p = 2$, 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.

Method	Median List	Best Cost	Best Time (s)	Average Cost	Average Number of Fitness Evaluations	Average Time (s)
GA+SA	183 583	6354	0.61	6354	8745	1.88
GA	178 368	6354	0.08	6354	9165	1.58
Exact	178 368	6354	37.05	6354	288420	34.71
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	65 277 445 485 585	3774	10.73	3774	124353	40.58
GA	104 266 386 646 656	3774	12.23	3774	348330	115.69
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	55 88 266 278 394 468 544 584 658 696	2479	199.40	2494.3	248802	149.88
GA	55 88 266 278 394 468 524 584 658 696	2479	263.28	2488.6	468939	294.80
Method	**Median List**	**Best Cost**	**Best Time (s)**	**Average Cost**	**Average Number of Fitness Evaluations**	**Average Time (s)**
GA+SA	50 63 77 174 184 189 240 247 253 357 366 389 488 494 521 581 648 683 710 717	1632	703.91	1639.5	540972	608.92
GA	29 56 63 149 174 240 253 267 357 365 391 438 493 520 528 581 648 683 717 730	1637	1640	1646	1394229	1696.89

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.

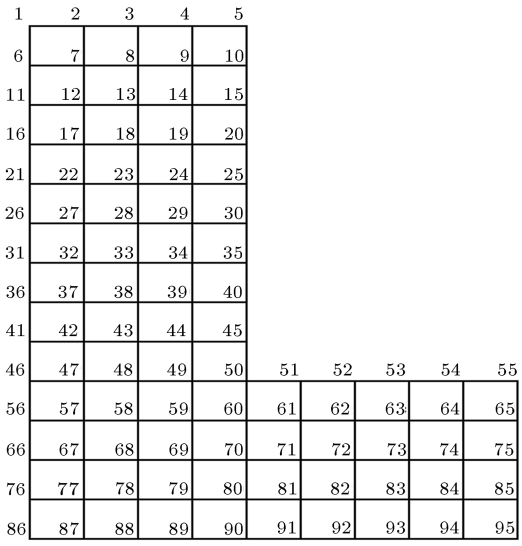


Figure 11. Example 3 with 95 nodes.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	
81	82	83	84	85											96	97	98	99	
86	87	88	89	90											101	102	103	104	
91	92	93	94	95											106	107	108	109	
111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	
131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	
171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	

Figure 12. Example 4 with 190 nodes and nodal weights of unity.

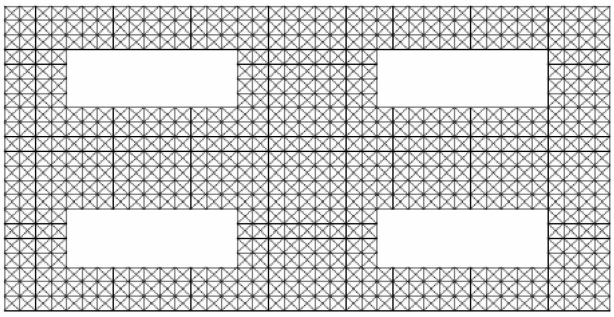


Figure 13. A graph with 760 nodes as the model of Example 5.

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

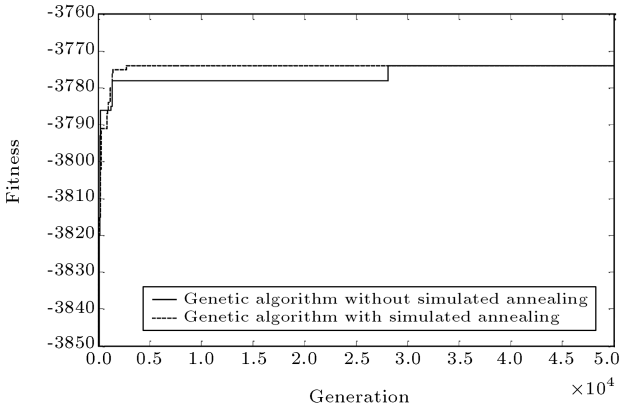


Figure 14. Sample convergence curves of GA and Hybrid GA+SA for $P = 5$ and $N = 760$.

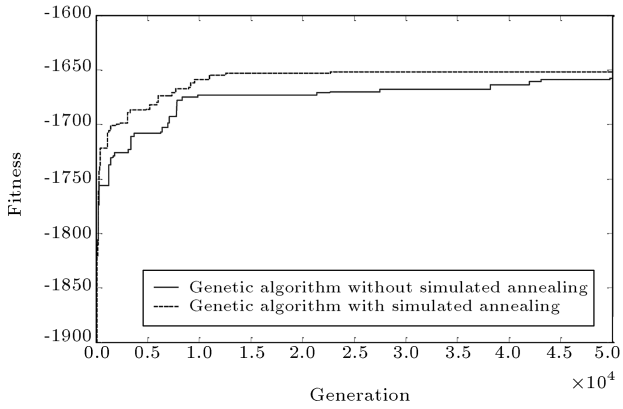


Figure 15. Sample convergence curves of GA and Hybrid GA+SA for $P = 20$ and $N = 760$.

also shows that t5he 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

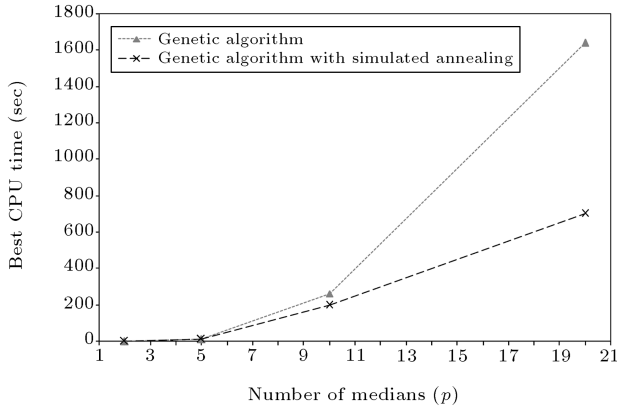


Figure 16. Comparison of algorithms efficiency of best achieved results for various number of medians in 760-node example.

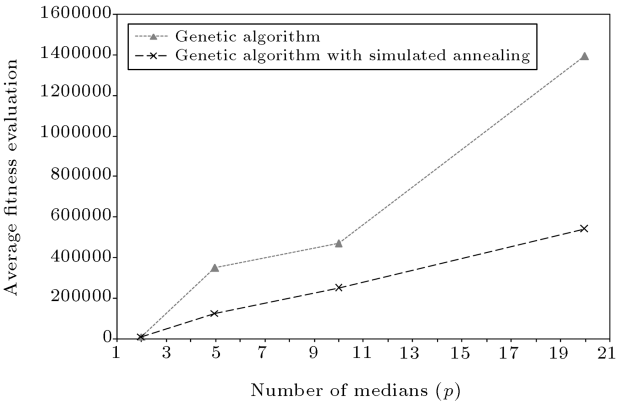


Figure 17. Comparison of algorithms efficiency in average number of fitness evaluations or various number of medians in 760-node example.

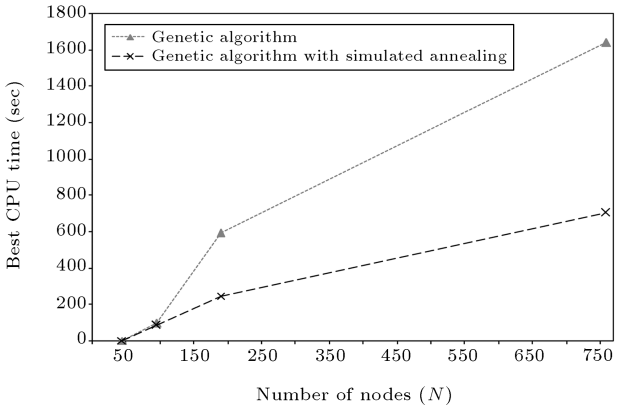


Figure 18. Comparison of algorithms efficiency for best achieved results and various number of nodes ($N = \{45, 95, 190, 760\}$).

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

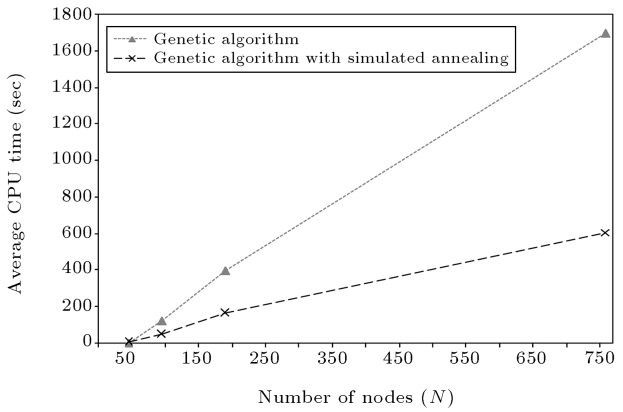


Figure 19. Comparison of algorithms efficiency in average CPU time results for various number of nodes.

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

1. Christofides, N., *Graph Theory: An Algorithmic Approach*, Academic Press, New York (1975).
2. Kaveh, A. and Shojaee, S. "Optimal domain decomposition via p-median methodology using ACO and hybrid ACGA", *Finite Elements in Analysis and Design*, **44**, pp. 505-512 (2008).
3. Khan, A.I. and Topping, B.H.V. "Parallel adaptive mesh generation", *Computing Systems in Engineering*, **2**, pp. 75-102 (1991).
4. Kaveh, A. and Davaran, A. "Spectral bisection of adaptive finite element meshes for parallel processing", *Computers and Structures*, **70**, pp. 315-324 (1999).
5. Kariv, O. and Hakimi, S.L. "An algorithmic approach to network location problems: Part 2. The p-medians", *SIAM Journal on Applied Mathematics*, **37**, pp. 539-560 (1969).
6. Papadimitriou, C., *Computational Complexity*, Addison-Wesley (1994).
7. Charikar, M. and Guha, S. "Improved combinatorial algorithms for the facility location and k-median problems", *40th Annual Symposium on Foundations of Computer Science*, pp. 378-388 (1999).
8. Osman, I. and Christofides, N. "Capacitated clustering problems by hybrid simulated annealing and tabu search", *International Transactions in Operational Research*, **1**(3), pp. 317-336 (1994).

9. McKendal, A.R. and Shang, J. "Hybrid ant systems for the dynamic facility layout problem", *Computers and Operations Research*, **33**(3), pp. 790-803 (2006).
10. Cheriyan, J. and Ravi, R. "Approximation algorithms for network problems", *Lecture Notes* (1998).
11. Mladenovic, N., Brimberg, J., Hansen, P. and Moreno-Perez, J. "The p-median problem: A survey of metaheuristic approaches", *European Journal of Operational Research*, **179**, pp. 927-939 (2007).
12. Kaveh, A. and Sharafi, P. "Ant colony optimization for finding medians of weighted graphs", *Engineering Computations*, **25**(2), pp. 102-120 (2008).
13. Mahfouz, S.Y., Toropov, V.V. and Westbrook, R.K. "Modification, tuning and testing of a GA for structural optimization problems", *Proceedings of 1st AMSO UK/ISSMO Conference on Engineering Design Optimization*, V. Toropov, Ed., pp. 271-278 (1999).
14. Yuan, B. and Gallagher, M. "A hybrid approach to parameter tuning in genetic algorithms", *Proceedings of IEEE Congress in Evolutionary Computation (CEC'05)*, **2**, pp. 1096-1103 (2005).
15. Holland, H.J., *Adaptation in Natural and Artificial Systems, an Introductory Analysis with Application to Biology, Control and Artificial Intelligence*, Ann Arbor, The University of Michigan Press (1975).
16. Kaveh, A. and Shahrouzi, M. "Adaptive selective pressure using hybrid evolutionary and ant strategies", *International Journal for Numerical Methods in Engineering*, **73**(4), pp. 544-563 (2007).
17. Kirkpatrick, S., Gelatt, C.D. Jr. and Vecchi, M.P. "Optimization by simulated annealing", *Science*, **220**(4598), pp. 671-680 (1983).
18. Kaveh, A. and Shahrouzi, M. "Simulated annealing and adaptive dynamic variable band mutation for structural optimization by genetic algorithms", *Asian Journal of Civil Engineering*, **7**(6), pp. 655-674 (2006).
19. Cormen, T.H., Leiserson, C.E. and Rivest, R.L., *Introduction to Algorithms*, The MIT Press, 2nd Ed., USA (2001).
20. Khumawala, B.M. "An efficient branch and bound algorithm for the warehouse location problem", *Management Science*, **18**(12), pp. 718-731 (1972).
21. Church, R.L. "BEAMR: An exact and approximate model for the p-median problem", *Computers & Operations Research*, **35**(2), pp. 417-426 (2008).
22. Correa, E.S., Steiner, M.T.A., Freitas, A.A. and Carnieri, C. "A genetic algorithm for the p-median problem", *Genetic and Evolutionary Computation Conference*, pp. 1268-1275 (2001).
23. Kaveh, A. and Shahrouzi, M. "Simultaneous topology and size optimization of structures by genetic algorithm using minimal length chromosome", *Engineering Computations*, **23**(6), pp. 644-674 (2006).

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 260 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 Shahrouzi 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 Shahrouzi is the author of 23 papers, 8 papers being published in international journals and 3 presented at international conferences.

Yaser Nasserifar 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.