Applied Mathematics & Information Sciences
An International Journal

A Learning based Evolutionary Approach for Minimization of Matrix Bandwidth Problem

Ayaz Isazadeh[1], *Habib Izadkhah*[2] *and A. H. Mokarram*[3]

[1] Department of Computer Science, Faculty of Mathematical Sciences, University of Tabriz, Tabriz, Iran
[2] Department of Computer Science, Faculty of Mathematical Sciences, University of Tabriz, Tabriz, Iran
[3] Department of Software Engineering, University of Daneshvaran, Tabriz, Iran

Received: Received December 22, 2010; Accepted June 20, 2011
Published online: 1 January 2012

Abstract: Nowadays, graphs and matrix have been used extensively in computing. In this paper an evolutionary approach to solve a problem related with the matrix called minimization of bandwidth problem is proposed. Due to difficulties to solve of this problem, using of evolutionary processing and especially genetic algorithm is efficient. In this paper by adding learning concepts such as penalties and rewards (Guidance) to the genetic algorithm, we obtained an efficient method for solving minimization of matrix bandwidth problem; so that in search process, the speed of finding the answer to the significantly increased. Obtained results of experiments on twenty sample matrices show the efficiency and speed of suggested method in comparison with other methods.

Keywords: enetic algorithm, Minimization of bandwidth, Graph

1. Introduction

Minimization the problem of matrix bandwidth is an old optimization problem suggested in about 1950 when building engineers were analyzing steel structures by using of computers and were putting none zero elements in as close as possible to the main diagonal by reordering and creation of an appropriate permutation of rows and columns of matrix and were using it to calculate the matrix inverse and determinant. Matrix bandwidth minimization problem in many areas are used. For example in solving large linear systems, the Gaussian elimination can be performed at the time duration of $O\left(nb^2\right)$ on matrix with bandwidth b (If $b \ll n$) which is much faster than the algorithm that performs at $O\left(n^3\right)$. This problem can have other applications in data storage, VLSI designing, finite element methods for approximation answers of partial differential equations and circuit design.

1.1. The Problem

Suppose that $A = \{a_{ij}\}_{n\times n}$ is the adjacency matrix of graph G when $a_{ij} = 1$ if the vertices i and j are adjacent otherwise it would be as $a_{ij} = 0$. The problem of minimizing bandwidth includes finding a permutation of rows and columns of matrix that keeps all the non-zero elements of A in a band that is as close as possible to the main diagonal in order to minimize the value of $\max\{|i-j|\} : a_{ij} \neq 0$.

Also, the problem of matrix bandwidth is defined as following: Suppose that $G = \{V, E\}$ is a graph, where $V\ (|V| = n)$ defines the set of vertices and E id the set of edges. One f labeling of graph G allocates $\{1, 2, \ldots, n\}$ to vertices of graph. In other words, f labeling is one as $f : V \to \{1, 2, \ldots, n\}$. If $f\ (v)$ to be allocated label to vertex V therefore bandwidth of vertex V shown as $B_f\ (v)$, is equal to the most difference between $f\ (v)$ and allocated labels to neighbor vertices of v. Therefore:

$B_f\ (v) = \max\{|f\ (v)| : (u, v) \in E\ \}$

Considering above definitions bandwidth of graph G in defined as a graph problem including finding of f labeling which minimizes $B_f\ (G)$ value.

The overall problem addressed in this paper is to discover whether there can be a practical and useful method for bandwidth minimization problem.

A solution to this problem is to pick (and if necessary modify) an existing formalism, or introduce a new one, with the following specific characteristics:

* Corresponding author:

1. Must minimize the bandwidth graph.
2. Must reduce the complexity of the optimization process and to minimize the running time of algorithm.

Our solution to this problem, stated clearly as our claim in Sec. 1.3, satisfies items 1.3.

1.2. Motivation and Background

Since there are $n!$ possible labeling for a graph with n vertices, the finding the minimum bandwidth of a graph is a complex optimization problem the need for precise bandwidth minimization problem has been widely recognized by the research community. A general taxonomy for bandwidth minimization problem algorithms has been reviewed and discussed by Chinn and Chvatalova [1]. Generally, the bandwidth minimization problem algorithms may be divided in two main classes; heuristic and deterministic algorithms. There are some known deterministic approaches to solve the bandwidth minimization problem [2,3]. However, to obtain optimal bandwidth of graph is a NP-complete problem [4]. Even simplified versions of the bandwidth minimization problem such as trees with a maximum degree of three are NP-complete [5].

Therefore, deterministic approaches are not suggested for bandwidth minimization problem with a relatively large number of nodes in graph [6,7,8]. Instead, an evolutionary approach such as genetic algorithms may be applied to solve such problems, efficiently.

Several evolutionary approach have been designed for bandwidth minimization problem, Examples the Cutchill–McKee algorithm [9], GPS [10], simulated annealing [11], An improved simulated annealing algorithm for bandwidth minimization [6], Tabu search [12], Heuristics for matrix bandwidth reduction [7], an improved version of the Cutchill-McKee algorithm [13], GA-SA [14], and GRASP-PR [15] Variable neighborhood search for bandwidth reduction[8]. These algorithms are highly successful in solving large instances.

1.3. The claim

In this paper a fast and efficient method by using improved genetic algorithm to solve the minimization of bandwidth problem is proposed. The evolutionary process of genetic algorithm is quickened using guidance (reward and penalize). For this purpose, beside evaluation of chromosomes, the genes are evaluated based on its effect on chromosome value. So, the most proper location for genes inside chromosomes and the most proper chromosome inside the primary population is gradually determined during the evolutionary process. This event led to quickening of evolutionary process and nearness of answers in different executions of new algorithm and also prevents from algorithm trapping in local optimal. The most important characteristic of this evolutionary process is its resistance against superficial changes in answers. In other words, a flexible balance is created between efficiency and stability. Thereby, it achieves a better bandwidth in comparison with the existing approaches and also obtains the result very swiftly.

In Fig.1.3, a pseudo code of our proposed algorithm is given. Different parts of this algorithm are further described in the following sections.

Step1:

-Create initial population as described in Section 2.2.
-Produce a quarter of the population randomly.
-Produce the rest of the population as follows:
  -Identify all the critical nodes on the graph.
- **For** i=1 **to** Population Size **Do**
  -Generate Random linear list of nodes.
  -**Repeat**
  -Select randomly a node amongst the nodes at the beginning of the linear list.
  -**If** the selected node is a the critical node **Then**
  -Find a non-critical vertex randomly and replace its number with the critical vertex and assign the vertex number to one of the genes.
  Else
    Assign the vertex number to one of the genes
  -Remove the selected node from the linear list.
  -**Until** the linear list of node is empty.

Step 2:

-**While** termination condition has not been reached **Do**
  -**For** each Chromosome in current population **Do**
  -Evaluate the fitness of each chromosome in population as described in Section 2.3.
  -Create intermediate generation as follows:
  -Add the fittest chromosome to the intermediate population.
  -**Repeat**
  -apply new proposed selection to select two chromosomes, described in Section 2.4.
  -apply the new crossover operator, described in Section 2.5.
  -apply the mutation operator, described in Section 2.5.
  -apply the reward and Penalize operator, described in Section 2.6.
  -**Until** the intermediate population size is completed.
  -Copy the intermediate population over current population.

Fig. 1. A new genetic bandwidth minimization algorithm

1.4. Paper outline

The remaining parts of this paper are organized as follows: In Section 2, a new algorithm for bandwidth minimization algorithm is presented. Section 3, practical results of implementing proposed algorithm is evaluated.

2. The proposed algorithm for solving the Minimization of Matrix Bandwidth problem

In this section a new algorithm is presented for solving the bandwidth minimization problem. In this algorithm the genetic evolutionary process is accelerated by using learning concept such as penalties and rewards. In the next subsections, the proposed new algorithm is studied in detail.

2.1. Chromosome representation

For example, consider the matrix shown in Fig. 1 which has six rows and columns. So that can be drawn as a graph with six vertices. Consider $\{1, 5, 6, 3, 2, 4\}$ permutations

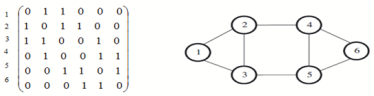

Figure 1 A sample of Matrix and Graph

of the matrix (graph) in Fig. 1. This permutation is shown in a chromosome in Fig. 2. In this representation the number of genes is equal to the number of graph vertices. For exact evaluation of genes and performing learning process, the internal states or internal memory is defined inside chromosome for each gene called gene depth, that in Fig. 2 equals with 5. The nodes on the circular are set of boundary states and other nodes are a set of internal states of chromosome genes. In each chromosomes after evaluation of gene fitness that choosing randomly, gene receives penalty or reward. Due to giving penalty or reward to a gene, the gene state changes in a set of status. If a gene to be in boundary state of a chromosome then getting of penalty leads to creation of a new permutation and if it gets reward then moves toward internal or fixed state. Rate of this operator should be low because it is a random search operator and if performs with high rate then leads to decreasing the algorithm efficiency. A set state of $\{1, 6, 11, 16, 21, 26\}$ are internal states and a set state of $\{5, 10, 15, 20, 25, 30\}$ are boundary states. Initially each labels attributed to the vertices are in the boundary state of the gene.

2.2. Initial Population

The evolutionary process of genetic algorithms usually starts from an initial population of randomly generated. The quality of the initial population affects the performance of the genetic algorithm. An initial population with suitable fitness values increased the speed of reaching an optimal solution, dramatically. Also, high diversity in the

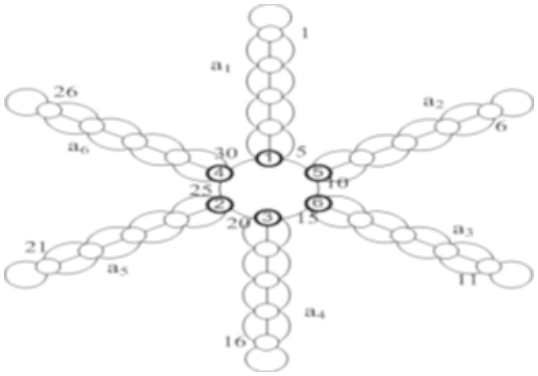

Figure 2 A sample of Matrix and Graph

population, prevent early convergence to a locally optimal solution. In order to maintain diversity, in our proposed algorithm, about a quarter of the initial population is produced randomly. To have a better fitness, a new approach is suggested to produce the rest of the initial population.

In order to produce initial population we generate them randomly then start to recognition and removing of critical vertices and replacing them with other vertices. To do this we first start to recognize critical vertices and then remove them as much as possible. Removing of critical vertices can decrease the amount of current $B_f(G)$ bandwidth. The set of critical vertices are defined as following:

$$C(f) = \{v : B_f(v) \geq \alpha B_f(G)\} \qquad 0 < \alpha < 1$$

n order to create a set of appropriate vertices interchangeable with each of $v \in C(f)$ vertices we define following formula:

$max(v) = max\{f(u) : u \in N(v)\}$
$min(v) = min\{f(u) : u \in N(v)\}$
$mid(v) = \lfloor \frac{max(v)+min(v)}{2} \rfloor$

In the above formula $N(v)$ is the set of vertices adjacent to v. Therefore we can define a set of appropriate vertices for replacing with v vertex as following:

$$N(v) = \{u : |mid(v) - f(u)| < |mid(v) - f(v)|\}$$

Which includes all u vertices with $f(u)$ label that their distance from $mid(v)$ is lesser than distance of $f(v)$ from $mid(v)$.

We arrange available vertices in the $N(v)$ according to $|mid(v) - f(u)|$ value in ascending mode. Then in the arranged sequences we replace $f(v)$ label from v vertex with $f(u)$ from u vertex until we obtain the improved answer. That is when $B_f(G)$ decreases.

2.3. Computing fitness of chromosomes

In the problem of minimization of graph bandwidth, the purpose is finding a labeling map like f in which $B_f(G)$ is

in minimum amount. Therefore fitness function calculates created bandwidth by f mapping.

$prucedureEvalFitness(f)$
$Begin$
$B_f(G) = |V|$
$For\ all\ v \in V do$
$B_f(v) = |V|$
$For\ all\ u\ that\ (u,v) \in E\ do$
$If\ |f(v) - f(u)| > B_f(v)\ then$
$B_f(v) = |f(v) - f(u)|$
$endif$
$endfor$
$if\ B_f(v) > B_f(G)\ then$
$B_f(G) = B_f(v)$
$endif$
$endfor$
$EndProcedure$

2.4. Selection Operator

Selection operator is used to create of new generation. Select mode in the algorithm according to the quality of answers (their possibility) is as following: If population has n numbers then for an individual with minimum target function the distance amount of $[0, 1)$ is assigned, $[1, 3)$ distance is assigned for a person with second minimum target function and finally $[n(n-1)/2, n(n+1)/2)]$ is assigned for an individual with maximum target function. Then a random number produces between 0 and $n(n+1)/2$. The distances in which the produce number is located determines selected individual.

2.5. Crossover operator and Mutation operator

In this method, two parental chromosomes are selected and two genes i and j are randomly chosen in one of the parental chromosomes. Then the same two genes are chosen in the other parental chromosome. Genes set with numbers between i and j are called replacement set. Then genes with same numbers are replaced in two replacement sets. Thereby, two chromosomes are produced which refer to children in two parental automata.

$prucedure\ Crossover(chromosome_1, chromosome_2)$
$Begin$
$Generate\ two\ random\ numbers\ r_1\ and\ r_2\ between\ 1\ to\ n$
$r_1 = Random * n;$
$r_2 = Random * n;$
$r_1 = Min(r_1, r_2);$
$r_2 = Max(r_1, r_2);$
$for\ i = r_1\ to\ r_2\ do$
$if(Fitness_i(chromosome_1) < Fitness_i(chromosome_2))\ then$
$j = gen\ of\ chromosome_2;$
$chromosome_2.gene(j) = chromosome_1.gene(i);$
$Swap(chromosome_2.gene(i), chromosome_2.gene(j));$
$end\ if$
$else$
$j = gen\ of\ chromosome_1;$
$chromosome_1.gene(j) = chromosome_2.gene(i);$
$Swap(chromosome_1.gene(i), chromosome_1.gene(j));$
$end\ else$
$end\ for$
$End\ Procedure$

Figure 3 Crossover Algorithm

$prucedure\ Mutation(chromosome)$
$Begin$
$i = Random * n;$
$j = Random * n;$
$Swap(chromosome.gene(i), chromosome.gene(j));$
$End\ Procedure$

Figure 4 Mutation Algorithm

2.6. Penalty and reward operator

In each chromosome after evaluation of the quality of each gene that chosen randomly, the reward or penalty will be given to the gene.

Suppose that a chromosome has $2N$ state of $\rho_1, \rho_2, ..., \rho_{2N}$, and includes $\{\alpha_1, \alpha_2\}$ two genes. States of $\rho_1, \rho_2, ..., \rho_N$, are related to α_1 and $\rho_{N+1}, \rho_{N+2}, ..., \rho_{2N}$ are related to the α_2 gene. Therefore: $G[\rho_i] = \begin{cases} \alpha_1 & i = 1, 2, \ldots, N \\ \alpha_2 & i = N+1, \ldots, 2N \end{cases}$ So that G is a mapping function.

If gene α_1 in the chromosome to be in state $(1 \le i \le N), \rho_i$ and if this gene gets penalty then the state changes as following: $\rho_i \Rightarrow \rho_{i+1}\ , (i = 1, 2, , N-1)$ $\rho_N = \rho_{2N}$ If the genes get the rewards then transferring of state performs as following: $\rho_i \Rightarrow \rho_{i-1}\ , (i = 1, 2, , N)$ $\rho_1 = \rho_1$ If gene α_2 to be in state $(N+1 \le i \le 2N), \rho_i$ in the chromosome and if it gets penalty then the state transfers performs as: $\rho_i \Rightarrow \rho_{i+1}\ , (i = N+1, N+2, , 2N-1)$ $\rho_{2N} = \rho_N$ If the genes get the rewards then transferring of state performs as following: $\rho_i \Rightarrow \rho_{i-1}\ , (i = N+2, N+3, , 2N)$ $\rho_{N+1} = \rho_{N+1}$

Due to giving penalty or reward to a gene, the gene state changes in a set of states. If a gene to be in boundary state of a chromosome then getting of penalty leads to creation

of a new permutation and if it gets reward then moves toward more internal states. Penalty or reward function in the suggested algorithm is so that we calculate bandwidth of one vertex and if the amount of bandwidth to be equal β with equal with current value of bandwidth then the reward will be paid to this vertex otherwise it will get penalty (β is an input parameter).

We first consider the probability value or quality of genes in each chromosomes of the population as $P_i(t) = \frac{1}{r}$ where r equals with the number of genes in the chromosomes. Depending on reward or penalty of one gene from the chromosome the probability of all genes in one chromosome in the population become updated.

Procedure : **reward** *and* **penalty algorithm**
Begin
for $i = 1$ **to** *size of population* **do**
begin
$u = Random * n;$ $//n = |V_G|$
Select *the* Gen_u *of the* $Chromosome_i$
if$(P(Gen_u) \leq reward\ and\ penalty\ rate)$ **then**
begin
if$B_f(u) \geq \beta\ x\ B_f(G)$ **then**
begin
$reward(u);$
$P_i(t+1) = P_i(t) + \sum_{i \neq j} \alpha P_j(t)$
$P_j(t+1) = P_j(t) - \alpha P_j(t) \qquad (\forall_j \neq i)$
end;
else
begin
$Penalize(u);$
$P_i(t+1) = P_i(t) - \sum_{i \neq j}^{r} \left(\frac{\beta}{r-1} - \beta P_j(t)\right)$
$P_j(t+1) = P_j(t) - P_j(t) + \frac{\beta}{r-1} - \beta P_j(t) \qquad (\forall_j \neq i)$
end;
end;
end
End

3. PRACTICAL RESULT

The proposed algorithm (LAGA) has been implemented in the Delphi programming language on an Intel P4 1.6 GHz machine. Test results and their analysis are classified in two parts: First part for the LAGA algorithm, the effect of parameters, including the number of generations, crossover and mutation rates, reward and penalty rates, were tested. In second part, proposed algorithm is compared with different algorithms. The comparison is firstly made by applying our algorithm to the graph used by other known algorithms.

At the beginning of the tests, we arrange applied parameters in the algorithm and after choosing of appropriate values perform comparable tests of new algorithms and other algorithms. In order to show the effect of parameters on algorithm, the matrix of bp_1000 from set of Harwell-Boeing Sparse Matrix, is used.

To test the effect of Population Size in LAGA, the number of generations was set to be 300, the crossover rate to be 1.0 and the mutation rate to be 0.02. The results are shown in the Fig. 5. Fig. 5 shows the effect of population size on the suggested algorithm. Fig. 5 shows that with increasing of population size the bandwidth decreases but on the other hand with increasing of population size, the time of algorithm increased. Therefore we consider population size as 30 in order to make a balance between results and speed of algorithm.

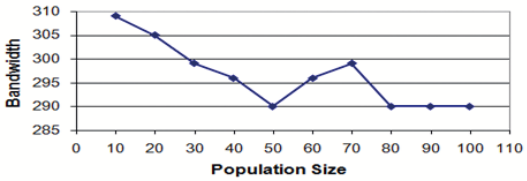

Figure 5 Effect of Population Size

In order to test the effect of depth on the suggested algorithm we select it from $\{1, 2, , 10\}$. By doing so the Fig. 6 obtains. As illustrated in Fig. 6 the appropriate depth is 3. In order to test the effect of crossover operator rate we choose it from $\{0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 1\}$. The result illustrated in Fig.7 As it can be seen the best results are obtained when the rate of crossover operator is about 0.8. Therefore we put the amount of crossover operator rate equal with 0.8.

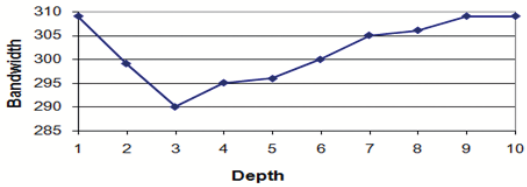

Figure 6 Effect of Depth

In order to test the effect of mutation operator rate we choose it from $\{0.02, 0.04, 0.05, 0.09, 0.2, 0.4\}$. The result illustrated in Fig. 8. As you can see the minimum amount of bandwidth is obtained when the rate of mutation operator is between 0.04 and 0.06. Therefore we put the amount of mutation operator rate equal with 0.05.

In order to test the effect of algorithm generations we perform the number of repeated algorithms started

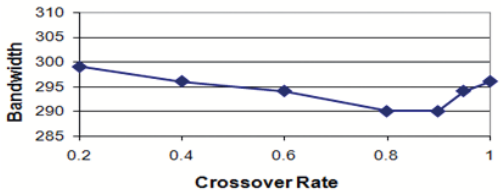

Figure 7 Effect of Crossover Rate

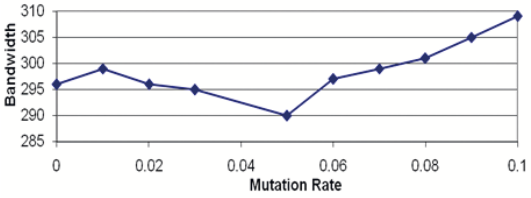

Figure 8 Effect of Mutation Rate

at 30 and then increase the number of repeated algorithms gradually. As it can be seen in 9 the graph bandwidths up to 70 generation decreases rapidly. Some tests

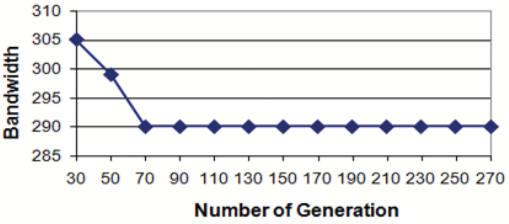

Figure 9 Effect of number of generations

performed on samples of Harwell-Boeing Sparse Matrix set. This collection includes two classifications: First classification includes 10 samples with number of vertices between 30 to 199 vertexes. Second classification includes 10 samples with number of vertices between 200 to 1000 vertexes. All these sample matrices are obtained from linear systems and other engineering and scientific problems. This collection can be obtained via http://math.nist.gov/ MatrixMarket/data/Harwell-Boeing http:// math.nist.gov/MatrixMarket/ data/Harwell-Boeing

In Table 1 and Table 2 the efficiency of suggested algorithm on some sample matrix from a set of Harwell-Boeing Sparse Matrix, are performed and compared with **GA-HC, GRASP-PR, TS** and **GPS** algorithms. Table 1 and Table 2 shows that the execution time of suggested algorithm is less than other algorithms. The new algorithm is an

Matrix	Best Result	GA-HC		GRASP-PR		TS		GPS		LAGA	
		$B_f(G)$	T	$B_f(G)$	T	$B_f(G)$	T	$B_f(G)$	T	$B_f(G)$	T
ash85	9	9	1	9	0.58	10	0.7	11	-	9	0.5
bcspwr01	5	5	0.45	5	0.06	5	0.1	7	-	5	0.3
bcsstk05	20	21	2.13	20	2.42	23	4.7	24	-	23	1.4
curtis54	10	10	0.84	10	0.45	10	0.7	14	-	10	0.6
gre__185	21	22	2.83	22	3.98	22	6.2	21	0.01	22	2.53
nos4	10	10	1.03	10	1.08	10	1.1	11	-	11	1
pores_1	7	7	0.5	7	0.17	7	0.3	8	-	7	0.23
will199	66	66	14.31	70	16.56	70	12	78	-	75	5
will57	7	7	0.89	7	0.23	7	0.4	7	0.01	7	0.23
west0132	33	33	3.66	36	6.61	36	3.4	39	0.01	39	2.9

$B_f(G)$= Bandwidth, T= Execution Time (Second)

Table 1: Performance comparison using 10 small instances from the Harwell-Boeing Sparse Matrix Collection

Matrix	Best Result	GA-HC		GRASP-PR		TS		GPS		LAGA	
		$B_f(G)$	T	$B_f(G)$	T	$B_f(G)$	T	$B_f(G)$	T	$B_f(G)$	T
bcspwr04	26	29	4.55	26	2.52	26	9.6	50	-	26	2
bp__1200	295	295	218.45	306	321.16	320	674.8	462	0.01	299	150
fs_541_1	270	270	23.7	270	9.42	270	54.4	529	5.59	270	8.23
gre__512	36	36	45.17	36	14.66	37	77.1	36	-	36	10.5
jpwh_991	94	99	100.84	98	30.25	94	312.6	147	0.01	94	45
mcfe	127	127	118.94	128	106.44	135	800.2	181	0.02	128	90.2
nnc666	45	46	57.38	48	27.97	45	138.5	64	0.01	48	14
str4	27	27	38.05	27	1.69	29	33	27	-	27	10
shl_400	240	240	50.92	247	31.38	243	188.3	425	0.01	243	15.59
west0381	153	153	120.45	159	56.63	164	113	287	0.02	159	48

$B_f(G)$= Bandwidth, T= Execution Time(Second)

Table 2: Performance comparison using 10 large instances from the Harwell-Boeing Sparse Matrix Collection

appropriate algorithm that the quality of its results is approximately close GRASP-PR, TS and GA-HC algorithms but LAGA algorithm is approximately 3 times faster especially in Table 2. According to GPS as the fastest algorithm suggested so far for solving the problem but its quality of results are too poor and considering the better quality results of the suggested algorithm than GPS, therefore it can be said that LAGA can be considered as an efficient algorithm which provides appropriate answers in a reasonable time. In other words this algorithm maintains the balance between quality and time.

4. CONCLUSION

Bandwidth minimization problem is NP-Hard. To solve this NP-Hard problem, evolutionary approaches such as genetic algorithms are quite effective. The work presented in this paper has established a quick and stable algorithm for the bandwidth minimization problem to achieve higher performance. We have presented a new and hybrid algorithm for solving the problem stated in Sec.1.1. Recall that our proposed solution is characterized as a new method, satisfying a list of specific characteristics: 1) we have given a precise method by combination of genetic algorithm and learning concept, also we reach to an efficient search method for solving the bandwidth minimization problem; so that, the speed of finding result increases

sensibly, 2) we present the self-reform, breeding, penalty and reward properties for our suggested combinational algorithm which these properties prevented from trapping in local optimal. In the future we are going to extended our experimentation, on a set of 113 well-known benchmark instances of the literature (Harwell-Boeing Sparse Matrix Collection), to more evaluate the performance of LAGA algorithm. In these experiments our algorithm has been compared with several state-of-the-art algorithms. In these tests the suggested algorithm will be compared statistically in terms of answers quality and the time required for finding the answer by some known algorithms.

References

[1] P.Chinn, J.Chvatalova, A.Dewdney, N.Gibbs, *The bandwidth problem for graphs and matrices*, a survey, Journal of Graph Theory 6 (3) (**1982**) 223-254.

[2] G.D.Corso and G.Manzini, *Finding exact solutions to the bandwidth minimization problem*, a survey, Computing 62 (3) (**1999**) 189-203.

[3] E.Gurari and I.Sudborough, *Improved dynamic programming algorithms for bandwidth minimization and the min-cut linear arrangement problem*, Journal of Algorithms 5 (**1984**) 531-546.

[4] C.Papadimitriou, *The NP-Completeness of the bandwidth minimization problem*, Journal on Computing 16 (**1976**) 263-270.

[5] M.Garey, R.Graham, D.Johnson and D.Knuth, *Complexity results for bandwidth minimization*, SIAM Journal on Applied Mathematics 34 (**1978**) 477-495.

[6] E.Rodriguez-Tello, J.Hao and J.Torres-Jimenez, *An improved simulated annealing algorithm for bandwidth minimization*, European Journal of Operational Research 185 (**2008**) 1319-1335

[7] A.Lim, B.Rodrigues, Fei Xiao, *Heuristics for matrix bandwidth reduction*, European Journal of Operational Research 174 (**2006**) 69-91

[8] N.Mladenovic, D.Urosevic, D.Perez-Brito and G.Garcia-Gonzalez, *Variable neighbourhood search for bandwidth reduction*,European Journal of Operational Research 200 (**2010**) 14-27.

[9] E.Cutchill and J.McKee, *Reducing the bandwidth of sparse symmetric matrices*, In: Proceedings of the ACM National Conference, New York, (**1969**). pp. 157-172.

[10] N.Gibbs, W. Poole and P.Stockmeyer, *An algorithm for reducing the bandwidth and profile of a sparse matrix*, SIAM Journal on Numerical Analysis 13 (**1976**) 235-251.

[11] G.Dueck and J.Jeffs, *A heuristic bandwidth reduction algorithm*, Journal of Combinatorial Mathematics and Computers 18 (**1995**) 97108.

[12] R.Marti, M.Laguna, F.Glover and V.Campos, *Reducing the bandwidth of a sparse matrix with Tabu search*, European Journal of Operational Research 135 (2) (**2001**) 211-220.

[13] A.Esposito, M.F.Catallano, F.Malucelli and L.Tarricone, *A new matrix bandwidth reduction algorithm*, Operations Research Letters 23 (**1999**) 99-107.

[14] A. Lim, B.Rodrigues, F.Xiao, *Integrated genetic algorithm with hill climbing for bandwidth minimization problem*, Lecture Notes in Computer Science 2724 (**2003**) 1594-1595.

[15] E.Pinana, I.Plana, V.Campos and R.Marti, *GRASP and path relinking for the matrix bandwidth minimization*, European Journal of Operational Research 153 (**2004**) 200-210.

Ayaz Isazadeh received a B.Sc. degree in Mathematics from the University of Tabriz in 1971, an M.S.Eng. degree in Electrical Engineering and Computer Science from Princeton University in 1978, and a Ph.D. degree in Computing and Information Science from Queens University, Ontario, Canada. He is a professor in the Department of Computer Science at the University of Tabriz.

Habib Izadkhah received the B.Sc. degree in Computer Science from University of PayamNour (Iran) in 2005, the M.Sc. degree in software engineering from Shabestar Islamic Azad university in 2008. He is currently a Ph.D. student in the Computer Science Department at University of Tabriz, Iran. His current researches focus on view based software engineering and reverse engineering.

Amir Hosseinzadeh received the B.Sc. degree in Computer Science from Shabestar Islamic Azad university (Iran) in 2005, the M.Sc. degree in software engineering from Shabestar Islamic Azad university in 2008. His current researches focus on view based software engineering and reverse engineering.