BANDWIDTH MINIMIZATION PROBLEM

Chen Wang, Chuan Xu, Abdel Lisser

10th *International Conference on MOdeling, Optimization and SIMlation - MOSIM14 - November 5-7-2014- Nancy - France "Toward circular Economy"*

Bandwidth Minimization Problem

C. WANG, C. XU, A. LISSER

LRI (Laboratoire de Recherche en Informatique), Université Paris-Sud 11
Bât 650 Université Paris-Sud 11, 91405 Orsay Cedex, France
Wang.Chen@lri.fr, Chuan@lri.fr, lisser@lri.fr

ABSTRACT : *Bandwidth minimization problem consists of finding a permutation of the rows and columns of a sparse matrix in order to keep the non-zero elements in a band that is as close as possible to the main diagonal. In the recent decades, meta-heuristics have become useful approaches for solving difficult combinatorial optimization problems. In this paper, we use three meta-heuristics to solve the bandwidth problem, including Simulated Annealing(SA), Tabu Search(TS) and Variable Neighborhood Search(VNS). We combine the local search with basic VNS to improve the efficiency of the algorithm and solve bandwidth minimization problem. By the experiment results of 47 benchmark instances, the running time of the algorithm is reduced compared with which of meta-heuristic from the literature.*

KEYWORDS : *bandwidth, meta-heuristic, running time.*

1 INTRODUCTION

Matrix bandwidth minimization problem is a well-known problem. This problem consists of finding a permutation of the rows and columns of a sparse matrix in order to keep the non-zero elements in a band that is as close as possible to the main diagonal.

The matrix bandwidth minimization problem originated in the 1950s when the structural engineers firstly analyzed the steel frameworks by computers : When we bring all the nonzero entries into a narrow band around the main diagonal and get an reordering matrix, the operations such as inversion and determinants will save time (Chinn P.Z. et al., 1982). Meanwhile, the graph bandwidth problem originated in 1962 at the Jet Propulsion Laboratory which studies on minimizing the maximum absolute errors of 6-bit picture codes represented by edge differences in a hypercube.

The main application of bandwidth minimization problem is to solve large linear systems. Gaussian elimination can be performed in $O(nb^2)$ time on matrices of dimension n and bandwidth b, which is faster than the forward $O(n^3)$ algorithm when b is smaller than n (Lim A. et al., 2006a). Besides, bandwidth minimization problem has a wide range of other applications. For examples, data storage, network survivability, industrial electromagnetic, saving large hypertext media (Berry M. et al., 1996), finite element methods, large-scale power transmission systems, circuit design, chemical kinetics, numerical geophysics (Pinana E. et al., 2004).

Because of the wide range of applications, the bandwidth minimization problem has generated a strong interest in developing algorithms for solving it since 1960s. Papadimitriou (Papadimitriou C.H., 1976) showed that the bandwidth minimization problem is NP-complete, which means it is not very likely that an algorithm can find the minimum matrix bandwidth in a polynomial time. In addition, Garey et al. (Garey M.R. et al., 1978) proved that the bandwidth minimization problem is NP-complete even if the input graph is a tree whose maximum vertex degree is 3. Therefore, except for the simplest cases, several heuristic algorithms have been proposed in the literature to try to find good quality solutions as fast as possible (Rodriguez-Tello E. et al., 2008). However, most of the proposed heuristic methods are specific or dedicated to a given problem, so recently more general algorithms are proposed which are called meta-heuristic (Loiola E.M. et al., 2007).

This paper will focus on meta-heuristic for solving bandwidth minimization problem, especially we use VNS framework to solve the problem. We combine the improved hill climbing method which is proposed in (Lim A. et al., 2006a), the reduced swap neighborhood presented in (Martí R. et al., 2001) and the shaking method proposed in (Mladenović N. et al., 2010) with the basic VNS for applying on bandwidth minimization problem. Section 2 concentrates on the two formulations of bandwidth minimization problem : matrix bandwidth minimization problem and graph bandwidth minimization problem, and an example shows the formulation specifically and the equivalence between the matrix and graph versions.

Section 3 discusses the literature for solving bandwidth minimization problem including exact algorithm, heuristic, and especially meta-heuristic. Section 4 introduces the basic VNS and describes the detail of each step of our VNS for solving bandwidth minimization problem. Section 5 concentrates on the computational experiments and compares the results of different three meta-heuristics which solve the bandwidth minimization problem. Section 6 concludes the paper.

2 PROBLEM FORMULATION

2.1 Matrix Bandwidth Minimization Problem

The matrix bandwidth minimization problem (MBMP) is defined as follows : Given a 0-1 sparse symmetric matrix $A = \{a_{ij}\}$, the bandwidth of matrix A is

$$B(A) = max\{|i-j| : a_{ij} \neq 0\} \tag{1}$$

Thus, the MBMP consists of permuting the rows and columns of matrix A to keep the non-zeros elements in a band that is as close as possible to the main diagonal, that is to minimize the bandwidth $B(A)$.

2.2 Graph Bandwidth Minimization Problem

The bandwidth minimization problem can be stated in the context of graph as follows : Let $G = (V, E)$ be a finite undirected graph, where V is the set of vertices and E is the set of edges, and a one to one function $f : V \rightarrow \{1, 2, ..., n\}$ is the labeling of its nodes, then the bandwidth of vertex v is defined as

$$B_f(v) = \max_{i:(i,j)\in E} \{|f(i) - f(j)|\} \tag{2}$$

and the bandwidth of G for f is defined as

$$B_f(G) = max\{|f(i) - f(j)| : (i, j) \in E\} \tag{3}$$

The bandwidth minimization problem for graphs is to find a labeling f for which minimizes the graph bandwidth, that is the $B_f(G)$ is minimum.

2.3 Equivalence between Graph and Matrix Versions

The bandwidth minimization problem for graph and matrix versions are equivalent. These two versions are interconvertible by transferring the given graph into an incidence matrix A (Lim A. et al., 2006b). Following is an example we present to show this equivalence.

Example. Given an undirected graph $G = (V, E)$ with $|V| = 5$ and the given labeling $f : f(v_1) = 3$, $f(v_2) = 1$, $f(v_3) = 2$, $f(v_4) = 5$, $f(v_5) = 4$. The original graph is given in Figure 1.

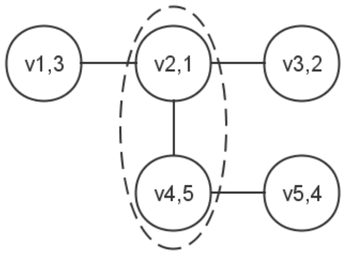

FIGURE 1 – Labeling f of graph G

Then the bandwidth of each vertex of the graph G under f are :

$$\begin{aligned}
B_f(v_1) &= \max\{|1-3|\} = 2\\
B_f(v_2) &= \max\{|3-1|, |2-1|, |5-1|\} = 4\\
B_f(v_3) &= \max\{|1-2|\} = 1\\
B_f(v_4) &= \max\{|1-5|, |4-5|\} = 4\\
B_f(v_5) &= \max\{|5-4|\} = 1
\end{aligned}$$

The bandwidth of the graph G under f is :

$$B_f(G) = \max_{v\in V} B_f(v) = \max\{2, 4, 1, 4, 1\} = 4$$

The adjacency matrix of the graph under labeling f is :

$$A(f) = \begin{pmatrix} 1 & 1 & 1 & 0 & 1\\ 1 & 1 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 1\\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

If we exchange the label of node v_1 with the label of node v_2, the resulting graph with new labeling f' is given in Figure 2.

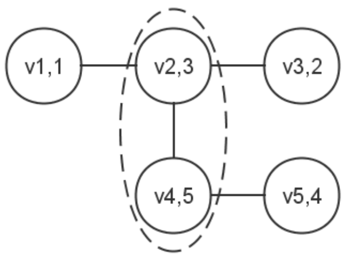

FIGURE 2 – Labeling f' of graph G

Currently, the bandwidth of each vertex under labeling f' is as follows :

$$B_{f'}(v_1) = \max\{|3-1|\} = 2$$
$$B_{f'}(v_2) = \max\{|1-3|, |2-3|, |5-3|\} = 2$$
$$B_{f'}(v_3) = \max\{|3-2|\} = 1$$
$$B_{f'}(v_4) = \max\{|3-5|, |4-5|\} = 2$$
$$B_{f'}(v_5) = \max\{|5-4|\} = 1$$

The graph bandwidth under $f^{'}$ is :

$$B_{f'}(G) = \max_{v \in V} B_{f'}(v) = \max\{2,2,1,2,1\} = 2$$

Hence, the bandwidth of graph has been reduced and the corresponding adjacency matrix $A(f^{'})$ is :

$$A(f^{'}) = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

For a graph with n vertices, the number of possible labeling is $n!$. The most direct method is to try all permutations and find which solution is the best. The running time for this method lies within a polynomial factor of $O(n!)$, so this approach becomes impractical even for small matrix which only have 10 vertices (Pop P. and Matei O., 2011).

3 SOLVING METHODS

Based on the literature, the algorithms of bandwidth minimization problem can be divided into two classes. The first one is exact algorithm. The second one is heuristic method, and recently meta-heuristic has been developed for this problem in order to obtain high quality solutions.

For the optimal labeling of vertices in the graph and optimal permutation of rows and columns in the matrix, the exact algorithms are mainly based on branch and bound search. Del Corso and Manzini (Del Corso G.M. and G. Manzini, 1999) was able to solve small and medium instances. Caprara and Salazar (Caprara A. and J.J. Salazar, 2005) extended the previous one by introducing tighter lower bounds, thus solved large size instances. However, for exact algorithm, the computational cost should be considered to obtain the optimal solution. Therefore, these methods can only be able to solve comparatively small size problems with a reasonable running time.

Heuristic refers to the technique which is based on experience, and it gives a solution which is not guaranteed to be optimal. However, the heuristic can quickly find a solution which is good enough for the combinatorial optimization problem. In 1969, the well-known Cuthill-McKee algorithm (Cuthill E. and J. McKee, 1969) appeared, which uses breadth first search to construct a level structure for graphs. Although the Cuthill-McKee algorithm was the most widely used method for bandwidth minimization problem during 1970s, it has several disadvantages. For example, the time consuming, the actual bandwidth might be less than the width of level structure (Chinn P.Z. et al., 1982). George (George J.A., 1971) proposed a reverse ordering for this problem. A few years later, Gibbs et al. (Gibbs N.E. et al., 1976) developed an algorithm known as GPS which is still based on the level structure. The experiment results showed that the GPS algorithm is comparable with the Cuthill-McKee algorithm while the time consuming is shorter.

Meta-heuristic is a technique which is more general than heuristic, because the heuristic method is usually specific for a given problem. Meta-heuristic can find a sufficiently good solution even optimal solution for the optimization problem with less computational assumptions. Therefore, in the past decades, much research has been focused on using meta-heuristic for solving complex bandwidth minimization problem.

In 2001, Martí et al. (Martí R. et al., 2001) solved the bandwidth minimization problem with tabu search. They introduced the reduced swap neighborhood based on the middle node, and a special candidate list strategy was used to increase the speed of move selection in a neighborhood. Extensive experiments showed that their tabu search outperforms the previous algorithms. Pinana et al. (Pinana E. et al., 2004) developed a Greedy Randomized Adaptive Search Procedure (GRASP) combined with a path relinking strategy for the bandwidth minimization problem. In 2006, Lim et al. (Lim A. et al., 2006a) presented a method combing the Genetic Algorithm (GA) and improved hill climbing to solve the bandwidth minimization problem. The improved hill climbing reduced the time complexity. Campos et al. (Campos V. et al., 2011) applied the Scatter Search (SS) for solving the bandwidth minimization problem. Rodriguez-Tello et al. (Rodriguez-Tello E. et al., 2008) proposed an improved simulated annealing algorithm. This method presented a representation of the solution, a rotation-based neighborhood and a new evaluation function, and the experiments showed the efficiency of the algorithm. Mladenović et al. (Mladenović N. et al., 2010) proposed a VNS method which combines several ideas from the literature for minimizing the bandwidth. The experiment results of 113 benchmark instances showed that the performance of the proposed VNS approach was better than all previous methods.

4 THE VNS APPROACH

Variable Neighborhood Search (VNS) is a meta-heuristic that is firstly proposed by Mladenović and Hansen (Mladenović N. and P. Hansen, 1997) in 1997. This meta-heuristic has been proved to be very useful for obtaining an approximate solution to optimization problems. VNS algorithm relies on the following three facts (Hansen P. et al., 2010) :

Fact1. The local optimum of a neighborhood structure is not necessarily a local optimal solution of another neighborhood structure.

Fact2. The global optimal solution is the local optimal solution for all possible neighborhood structure.

Fact3. For a lot of problems, the local optimums of several neighborhood structures are close to each other.

Therefore, VNS is an algorithm that systematically changes the set of neighborhood structure to expand the search range and obtain the local optimal solution until the best solution is found. The basic VNS algorithm includes three processes : shaking, local search and neighborhood change. Shaking is trying to jump out the current local optimum and find a new local optimal solution, while making the local optimal solution be closer to the global optimal solution. Local search is used to find the local optimal solution in order to improve search accuracy. Move or not means the neighborhood change which provides an iterative method and a stopping criterion. The detail of the algorithm for solving bandwidth minimization problem is described as follows.

4.1 Initial solution

The quality of initial solution will directly affect the performance of the algorithm, because a good initial solution can guarantee the algorithm to obtain the global optimal solution or near-optimal solution within a short time.

A good initial solution can be generated by a level structure procedure which using breadth first search (BFS). The idea is that adjacent vertices should have close labels. A level structure of a graph is denoted by $L(G)$, and it is a partition of the vertices into levels $L_1, L_2, ..., L_k$ which satisfy the following conditions (Martí R. et al., 2001) :

(1) vertices adjacent to a vertex in level L_1 are either in L_1 or L_2 ;

(2) vertices adjacent to vertex in level L_k are either in L_k or L_{k-1} ;

(3) vertices adjacent to vertex in level L_i (for $1 < i < k$) are either in L_{i-1}, L_i or L_{i+1}.

According to this, reasonable good solutions can be obtained. Therefore, initial solutions are generated by applying BFS with random selection of the starting vertex, and different starting vertices will provide different initial solutions. For example, for the matrix A, if we start from the vertex $v3$, the bandwidth decreases to 3. If we choose vertex $v2$ as the first label, the bandwidth is 2. Figure 3 and 4 show the examples of initial solution. According to the level structure, all the initial solutions are better than the original assignment. Obviously, the bandwidth obtained by this method can not be worse than the maximum bandwidth of the graph, because the adjacent vertices are assigned with sequential numbers. BFS method gives an upper bound of good quality solution.

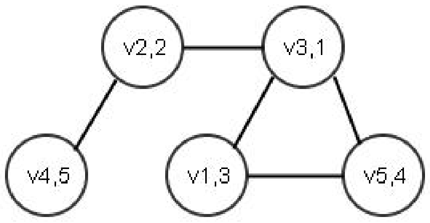

FIGURE 3 – $v3$ is the first label vertex

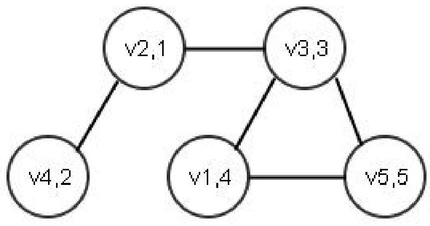

FIGURE 4 – $v2$ is the first label vertex

4.2 Shaking

A labeling f' is in the k_{th} neighborhood of the labeling f, that is, there are $k+1$ different labels between f and f'. More precisely, the distance ρ between any two solutions f and f' is defined as :

$$\rho(f, f') = \sum_{i=1}^{n} \eta(i) - 1, \eta(i) = \begin{cases} 1 & f(i) = f'(i) \\ 0 & f(i) \neq f'(i) \end{cases} \quad (4)$$

For example, the label f of Figure 3 is : $f = (3,2,1,5,4)$, and the label f' of Figure 4 is : $f' = (4,1,3,2,5)$, thus the distance between f and f' is 4. In order to choose the vertices to swap their labels, two definitions are added :

$$f_{max}(v) = max\{f(u), u \in N(v)\} \quad (5)$$

$$f_{min}(v) = min\{f(u), u \in N(v)\} \quad (6)$$

$f_{max}(v)$ indicates the maximum label of the adjacent vertex to vertex v, and $f_{min}(v)$ is the minimum label. For Figure 4, $f_{max}(v2) = 3$ and $f_{min}(v2) = 2$.

Firstly, a vertex set $K \subseteq V$ is defined whose cardinality is larger than k. Then a vertex u is chosen randomly from the set K and its critical vertex is also found. Next, a vertex w will be selected according to the conditions : $\max\{f_{max}(w) - f(v), f(v) - f_{min}(w)\}$ is minimum and $f_{min}(u) \leq f(w) \leq f_{max}(u)$. Finally the label of vertex v is replaced by vertex w.

In the following pseudo code, the shaking process can be presented as :

Algorithm 1 Shaking (k, f)

```
Initialization:
    Let $K = \{v | B_f(v) \geq B'\}$, $B'$ is chosen such that $|K| \geq k$;
Iteration:
 1: for $i = 1$ to $k$ do
 2:     $u \leftarrow$ RandomInt $(1, |K|)$;
 3:     $v \leftarrow$ such that $|f(u) - f(v)| = B_f(u)$;
 4:     if $(u, v) \in E$ then
 5:         $w \leftarrow \arg\min_w\{\max\{f_{max}(w) - f(v), f(v) - f_{min}(w)\} | f_{min}(u) \leq f(w) \leq f_{max}(u)\}$;
 6:         $swap(f(u), f(v))$
 7:     end if
 8: end for
```

4.3 Local search

We use the local search which is proposed in (Martí R. et al., 2001) to construct a set of suitable swapping vertices. The best labeling for current vertex v is defined as :

$$mid(v) = [\frac{max(v) + min(v)}{2}] \quad (7)$$

Then the set of suitable swapping vertices for vertex v is :

$$N^{'}(v) = \{u : |mid(v) - f(u)| < |mid(v) - f(v)|\} \quad (8)$$

According to the swapping vertices set $N^{'}(v)$, swapping the label of the current critical vertex v with the vertex $u \in N^{'}(v)$ is tried one by one in ascending value of $|mid(v) - f(u)|$ until the solution is improved (Lim A. et al., 2006a). Besides, if the bandwidth of the graph is not reduced, but the number of critical edges (critical edge means the bandwidth of the vertices connected with the edge is equal to the graph bandwidth $B_f(v) = B_f(G)$) is reduced, this condition can also be seen as the solution is improved. The local search procedure is given in Algorithm 2.

Algorithm 2 Local Search (f)

```
 1: while $CanImprove$ do
 2:     $CanImprove = False$;
 3:     for $v = 1$ to $n$ do
 4:         if $B_f(v) = B_f(G)$ then
 5:             for all $u$ such that $u \in N'(v)$ do
 6:                 swap $(f(v), f(u))$ and update $(B_f(w), B_f(G)), \forall w \in (N(v) \cup N(u))$;
 7:                 if number of critical edges reduced then
 8:                     $CanImprove = True$;
 9:                     $break$;
10:                 end if
11:                 swap $(f(v), f(u))$ and update $(B_f(w), B_f(G)), \forall w \in (N(v) \cup N(u))$;
12:             end for
13:         end if
14:     end for
15: end while
```

4.4 Move or not

After finding the local optimal solution, we must decide whether the current solution f is replaced by the new solution f'. The following three cases are considered : 1. $B_{f'}(G) < B_f(G)$: If the bandwidth of new solution is better than current solution, it is easy to determine the move. 2. $|V_c(f')| < |V_c(f)|$: If the bandwidth does not change, that is, $B_{f'}(G) = B_f(G)$, we compare the number of critical vertex for current and new solution to see if $|V_c(f')|$ is reduced. 3. $\rho(f', f) > \alpha$: If the two cases above are not satisfied, we compare these two solutions with a distance α which is a coefficient given by the user. The detail is presented in the Algorithm 3.

Algorithm 3 Move (f, f', α)

```
 1: $Move \leftarrow False$;
 2: if $B_{f'}(G) < B_f(G)$ then
 3:     $Move \leftarrow True$;
 4: else
 5:     if $B_{f'}(G) = B_f(G)$ then
 6:         if $|V_c(f')| < |V_c(f)|$ or $\rho(f', f) > \alpha$ then
 7:             $Move \leftarrow True$;
 8:         end if
 9:     end if
10: end if
```

Thus, the pseudo code of our VNS is presented in Algorithm 4.

5 NUMERICAL RESULTS

In order to evaluate the performance of the algorithm, we compare the solution and running time of our VNS with other two algorithms from the literature : Simulated Annealing (SA) (Torres-Jimenez J.

				VNS	Standard	Our	VNS	Simulate	Annealing	Tabu	Search
Instance	n	LB	Best	value	CPU	value	CPU	value	CPU	value	CPU
arc130	130	63	63	63	0.02	63	1.73	65	9.14	65	19.34
bcspwr01	39	5	5	5	0.42	6	0.00	5	0.07	5	0.02
bcspwr02	49	7	7	7	0.24	9	0.00	5	0.16	5	0.05
bcspwr03	118	9	10	10	1.44	14	0.03	13	0.37	10	1.84
bcsstk01	48	16	16	16	0.29	16	0.02	17	0.40	16	0.18
bcsstk04	132	36	37	37	0.04	38	1.89	41	28.34	39	15.9
can_144	144	13	13	13	0.23	14	0.11	15	1.70	13	7.24
can_161	161	18	18	18	0.48	24	0.52	24	1.98	21	7.68
fs_183_1	183	52	60	60	14.25	64	4.51	68	5.04	64	43.59
gent113	104	25	27	27	2.13	31	0.42	28	1.58	28	2.60
impcol_b	59	19	20	20	0.14	21	0.08	21	0.80	21	0.29
impcol_c	137	23	30	30	9.81	36	0.28	36	0.92	33	4.90
lund_a	147	19	23	23	0.02	23	0.02	23	9.59	23	10.90
lund_b	147	23	23	23	0.01	23	0.28	23	9.41	23	10.50
nos1	158	3	3	3	0.00	5	3.30	6	1.21	4	15.1
nos4	100	10	10	10	0.89	11	0.03	12	0.63	10	0.89
west0132	132	23	32	32	42.71	37	1.04	35	0.17	37	6.70
west0156	156	33	36	36	12.73	44	0.84	40	0.89	39	16.10
west0167	167	31	34	34	69.28	40	1.47	35	2.48	36	11.35
will199	199	55	65	65	11.28	76	11.32	53	3.37	53	51.75
will57	57	6	6	6	1.25	7	0.01	8	0.32	8	0.14
Average		23.28	25.61	25.61	7.98	28.67	1.33	27.29	3.74	26.33	10.81
Gap		9.10%		0%		11.95%		6.56%		2.81%	

TABLE 1 – Result of small dimension matrix

Algorithm 4 VNS $(A, k_{min}, k_{max}, k_{step}, \alpha)$

```
Initialization:
 1: $B^* \leftarrow \infty$ ;$t \leftarrow 0$ ;
 2: $i_{max} = Int((k_{max} - kmin)/k_{step}))$ ;
 3: $f \leftarrow InitSol(f)$ ;$f \leftarrow LocalSearch(f)$; ;
 4: $i \leftarrow 0$ ;$k \leftarrow k_{min}$ ;
 5: while $i \leq i_{max}$ do
 6:    $f' \leftarrow Shaking(f, k)$ ;
 7:    $f' \leftarrow LocalSearch(f')$ ;
 8:    if $Move(f, f', \alpha)$ then
 9:       $f \leftarrow f'$ ;$k \leftarrow k_{min}$ ;$i \leftarrow 0$ ;
10:    else
11:       $k \leftarrow k + k_{step}$; $i \leftarrow i + 1$ ;
12:    end if
13: end while
```

and E. Rodriguez-Tello, 2000) and Tabu Search (TS) (Martí R. et al., 2001). We tested 47 instances from the Harwell-Boeing Sparse Matrix Collection which are divided into two sets : the first set includes 21 instances (the dimension of the matrix ranging from 30 to 199) and the second set consists of 26 instances (the dimension of the matrix ranging from 200 to 1000). First, we transfer the matrix into the graph considering the incidence matrix, then we implement the algorithm with a graph formulation. Because the solution and running time of different algorithms are obtained from different computers, in order to compare the performance of these methods, we resume the experiment of different methods with our computer according to the literature description.

Table 1 and 2 summarize the result of different algorithms with 47 instances. For the instances, the algorithms are implemented in C and compiled with Microsoft Visual C++ 6.0, and the program was run with a Intel I7 at 2 GHz with 4 GB of RAM. In Tables 1 and 2, the first column indicates the name of the instance, the second column shows the size of the matrix. The third column presents the lower bound of the bandwidth obtained by the literature, and the fourth column shows the best solution of the bandwidth minimization problem. Then, the column of VNS standard presents the results from the literature (Mladenović N. et al., 2010). Value is the bandwidth and CPU is the running time of the algorithm. The other three columns are the results of our VNS, SA and TS. The last two rows show the average value and the running time of each method, and the gap we compute as $Gap = \frac{|v_{opt} - v_{best}|}{v_{best}} \times 100\%$ where v_{opt} is the optimal value of each method, and v_{best} is the best value which is showed in the fourth column.

For each instance, we test 10 times, and the best result is shown in Tables 1 and 2. For our VNS, we define the parameters as follows : $k_{min} = 2, k_{step} = 3, k_{max} = n/2, \alpha = 10$.

According to the result, our VNS does not work as well as in the literature (Torres-Jimenez J. and E. Rodriguez-Tello, 2000) (Martí R. et al., 2001) (Mladenović N. et al., 2010), but compared with the size of the matrix, we have significantly decreased the bandwidth, i.e., improved the quality of the upper bounds. Our VNS offers an advantage of the CPU time. Especially for large size matrices, it can solve the problem in a shorter time.

6 CONCLUSION

Bandwidth minimization problem, especially for the large size matrix is challenging because it is difficult to solve. Meta-heuristic is an effective approach to solve such optimization problems with few assumptions. In this work, we focus on using meta-heuristic to solve the bandwidth problem for sparse matrices. We transfer the matrix problem into a graph problem and apply variable neighborhood search (VNS) to solve it. By combining the improved local search with the basic VNS and defining the parameters which influent the neighborhood change, the experiment results show that our VNS is competitive with the state of art from the result quality point of view, and both for the small and large instances, our VNS outper-

				VNS	Standard	Our	VNS	Simulate	Annealing	Tabu	Search
Instance	n	LB	Best	value	CPU	value	CPU	value	CPU	value	CPU
ash292	292	16	19	19	39.35	27	1.10	24	4.63	19	9.61
bcspwr04	274	23	24	24	33.52	37	2.33	45	2.36	39	6.60
bcspwr05	443	25	27	27	28.55	56	3.34	54	2.90	39	17.14
bcsstk06	420	38	45	45	208.9	47	7.77	47	85.20	47	44.87
bcsstk19	817	13	14	14	199.34	18	33.22	28	52.10	25	280.30
bcsstk20	467	8	13	13	52.13	17	6.31	14	13.00	17	24.47
bcsstm07	520	37	45	45	208.90	66	23.02	57	74.70	47	43.02
can_445	445	46	52	52	119.68	77	16.67	61	14.60	54	75.29
can_715	715	54	72	72	192.68	119	151.56	88	62.05	87	229.73
can_838	838	75	86	86	402.23	107	61.72	104	148.25	99	284.48
dwt_209	209	21	23	23	25.30	33	0.62	30	3.82	28	6.12
dwt_221	221	12	13	13	23.88	17	0.29	20	2.41	15	5.36
dwt_245	245	21	23	23	25.3	31	0.58	23	1.98	18	9.94
dwt_310	310	11	12	12	11.45	16	5.92	20	5.17	12	23.52
dwt_361	361	14	14	14	7.22	18	11.30	22	9.30	16	15.37
dwt_419	419	23	25	25	69.21	30	2.84	45	26.56	42	33.81
dwt_503	503	29	41	41	174.40	63	13.81	56	93.10	55	228.24
dwt_592	592	22	29	29	111.32	34	8.18	53	47.70	50	84.75
dwt_878	878	23	25	25	111.32	37	23.01	41	40.20	34	300.05
dwt_918	918	27	32	32	223.19	53	53.51	55	165.20	52	180.50
plat362	362	29	34	34	179.34	45	6.91	39	84.24	36	38.44
plskz362	362	15	18	18	22.29	21	4.61	21	8.85	20	14.45
str_0	363	87	116	117	43.36	139	180.11	123	70.12	125	90.25
str_200	363	90	125	125	38.27	150	47.06	133	118.50	144	85.08
west0381	381	119	151	153	66.59	181	20.00	164	53.97	171	84.56
west0479	479	84	121	121	38.50	173	350.87	130	43.90	137	72.32
Average		37.53	45.75	45.82	108.80	61.14	39.27	57.58	47.49	54.92	88.01
Gap		17.96%		0.15%		33.63%		25.67%		19.86%	

TABLE 2 – Result of large dimension matrix

forms the state of art from CPU time point of view. For the future work, on one hand, we could further improve the result quality of our algorithm with considering to add a restart in the program so that it does not end early and may gain better solution. On the other hand, because of the reduced running time of our VNS, we can use this algorithm to solve very large size instances, i.e., matrices with more than 10,000×10,000.

ACKNOWLEDGMENTS

The first author is grateful for the financial support given by the China Scholarship Council.

REFERENCES

Berry M. W., B. Hendrickson, P. Raghavan, 1996. Sparse matrix reordering schemes for browsing hypertext. *Lectures in Applied Mathematics (LAM)*, American Mathematical Society, 32, p. 99-123.

Campos V., E. Pinana, R. Martí, 2011. Adaptive memory programming for matrix bandwidth minimization. *Annals of Operations Research*, 183(1), p. 7-23.

Caprara A. and J. J. Salazar, 2005. Laying out sparse graphs with provably minimum bandwidth. *INFORMS Journal on Computing*, 17(3), p. 356-373.

Chinn P. Z., J. Chvátalová, A. K. Dewdney, N. E. Gibbs, 1982. The bandwidth problem for graphs and matrices-A survey. *Graph Theory*, 6(3), p. 223-254.

Cuthill E. and J. McKee, 1969. Reducing the bandwidth of sparse symmetric matrices, *Proceedings of the ACM National Conference, Association for Computing Machinery*, New York, p. 157-172.

Del Corso G. M. and G. Manzini, 1999. Finding exact solutions to the bandwidth minimization problem. *Computing*, 62(3), p. 189-203.

Garey M. R., R. L. Graham, D. S. Johnson, D. E. Knuth, 1978. Complexity results for band-width minimization. *SIAM Journal on Applied Mathematics*, 34(3), p. 477-495.

George J. A., 1971. Computer implementation of the finite element method. *Stanford University California, Department of Computer Science.*

Gibbs N. E., W. G. Poole, P. K. Stockmeyer, 1976. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, 13(2), p. 236-250.

Hansen P., N. Mladenović, J. A. M. Pérez, 2010. Variable neighborhood search : methods and applications. *Annals of Operations Research*, 175(1), p. 367-407.

Lim A., B. Rodriguez, F. Xiao, 2006a. Heuristics for matrix bandwidth reduction. *European Journal of Operational Research*, 174, p. 69-91.

Lim A., J. Lin, B. Rodrigues, F. Xiao, 2006b. Ant colony optimization with hill climbing for the bandwidth minimization problem. *Applied Soft Computing*, 6(2), p. 180-188.

Loiola E. M. , N. M. M. Abreu, P. O. Boaventura-Netto, P. Hahn, T. Querido, 2007. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2), p. 657-690.

Martí R., M. Laguna, F. Glover, V. Campos, 2001. Reducing the bandwidth of a sparse matrix with tabu search, *European Journal of Operational Research*, 135(2), p. 450-459.

Mladenović N. and P. Hansen, 1997. Variable neighborhood search. *Computers & Operations Research*, 24(11), p. 1097-1100.

Mladenović N., D. Urosevic, D. Pérez-Brito, C. G. García-González, 2010. Variable neighbourhood search for bandwidth reduction. *European Journal of Operational Research*, 200(1), p. 14-27.

Papadimitriou C. H., 1976. The NP-completeness of the bandwidth minimization problem. *Computing*, 16(3), p. 263-270.

Pinana E., I. Plana, V. Campos, R. Marti, 2004. GRASP and path relinking for the matrix bandwidth minimization. *European Journal of Operational Research*, 153(1), p. 200-210.

Pop P. and O. Matei, 2011. Reducing the bandwidth of a sparse matrix with genetic programming. http ://www.researchgate.net/publication/221053552_An_Improved_Heuristic_for_the_Bandwidth_Minimization_Based_on_Genetic_Programming/file/79e41505af6aac067c.pdf.

Rodriguez-Tello E., J. K. Hao, J. Torres-Jimenez, 2008. An improved simulated annealing algorithm for bandwidth minimization. *European Journal of Operational Research*, 185(3), p. 1319-1335.

Torres-Jimenez J. and E. Rodriguez-Tello, 2000. A new measure for the bandwidth minimization problem, *Advances in Artificial Intelligence*, Springer Berlin Heidelberg, p. 477-486.