A New Evolutionary Algorithm with Level_NodeSwap for Bandwidth Minimization Problem

Kirti Varshney
M.Sc. Mathematics
Dayalbagh Educational Institute
Agra.
Contents:

• Bandwidth Minimization Problem [BMP]
• Applications of BMP
• Existing Approaches
• Proposed Technique
• Computational Results
• Conclusion
Basic Notations:

Graph : \( G = ( V, E ) \) with

Vertex Set : \( V = ( v_1, v_2, \ldots, v_n ) \) and

Edge Set : \( E = ( e_1, e_2, \ldots, e_m ) \) such that

Each undirected edge \( e_k \) is identified with an unordered pair of vertices \( ( v_i, v_j ) \) and is denoted by \( v_i v_j \).

A graph is finite if \( V \) and \( E \) are finite sets.

A loop is an edge \( v_i v_i \).

The set of neighbors of \( u \in V(G) \) is denoted as:

\[ N(u) = \{ v \in V(G) : uv \in E(G) \} . \]
What is Graph Layout?

A layout or simply a labeling of vertices, of an undirected graph $G = (V, E)$ with $n = |V|$ vertices, is a bijective function:

$$\varphi : V \rightarrow [n] = \{1, \ldots, n\}$$

- Unless otherwise mentioned, graphs are assumed to be finite, undirected and without loops.
- Length of an edge $uv$ on $\varphi$:

$$\lambda(uv, \varphi, G) = |\varphi(u) - \varphi(v)|, \quad uv \in V(G)$$
What is Bandwidth Minimization Problem [BMP]?

Bandwidth of graph $G(V, E)$: $BW(\phi, G) = \max_{uv \in E} \lambda(uv, \phi, G)$

Bandwidth minimization problem: “Given a graph $G = (V, E)$ find a layout $\phi^* \in \Phi(G)$ such that $BW(\phi^*, G) = \min BW(G)$”.

• BMP is a classic combinatorial optimization problem, originated in 1950’s.

• The problem has been proved to be NP-complete by Papadimitriou. Even if the input graph is a tree whose max vertex degree is 3.
Example: Labeling of vertices of graph

Layout: ABECDFGH
BW = 3 (max edge length corresponding to given layout)

Layout: AEBFDCHG
BW = 4 (max edge length corresponding to given layout)
Applications of BMP:

- To solve non-singular system of linear algebraic equations of the form $Ax = b$. The preprocessing of $A$ to reduce its bandwidth results in substantial saving on the computational effort associated with solving the system of equations.
- Finite element methods for approximating solution of partial differential equations.
- Large-scale power transmission systems
- Circuit design
- Hypertext layout
- Chemical kinetics
- Numerical geophysics
- Data-storage
- VLSI design and network survivability
Existing Approaches for BMP:

- Cuthill_McKee [CM]
- Reverse Cuthill_McKee [RCM]
- Gibbs, Poole, Stock, Meyer [GPS]
- Wonder Bandwidth Reduction Algorithm [WBRA]
- Node Shift Method
- Particle Swarm Optimization
- Node Centroid Adjustment Method with Hill Climbing [NCHC] etc.
Proposed Technique:

A new Technique, Evolutionary Algorithm with Level NodeSwap [EALNS] based on level structure of graph is proposed, which gives better solution quality in considerably less computational time.

What are Evolutionary Algorithms [EA]?

• Natural evolution is a population based optimization process, based on the principle of ‘survival of the fittest’.
• EA’s use population of candidate solutions in each iteration, instead of single solution.
• It is likely that expected EA solutions may be a global solution.
Level Structure:

- A level structure is a partition of vertices into levels $L_1, L_2, \ldots, L_k$, which have the following features:

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 in 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 $L_{i+1}$.

Given a level structure, $L$, the minimum bandwidth, $BW(\phi, G)$, when vertices are labeled sequentially by levels, is bounded as in the following range:

$$|L| \leq BW(\phi, G) \leq 2|L| - 1$$

where $|L|$ is the cardinality of the largest level, $L$, with the most vertices in the level structure.
Algorithm 1: Evolutionary Algorithm with Level_NodeSwap [EALNS]

1. Initialize population size(PS), max_iter;
2. Set iter = 0;
3. Create initial population $P(0)$, by RBFS; (The population constitutes parent strings: Parent_Str)
4. Evaluate $P(0)$;
5. While (iter $\leq$ max_iter) or not Termination Conditions() do
   6. iter = iter + 1;
   7. Apply Tournament Operator to produce new population $P(iter)$;
8. for i = 1 to PS do
   9. Level_NodeSwap (Parent_Str (i), New_Str);
10. Acceptance_Criteria (Parent_Str(i), New_Str);
11. end for
12. $P(iter)$ is updated with the parent strings.
13. end while
Random Breadth First Search [RBFS]:

In RBFS, the breadth first search (BFS) tree is constructed by picking the neighbors of the vertices in a random order (see figure 1.)

Example: Starting RBFS from the vertex A, two of the possible RBFS sequences are ABECDFGH and AEBFDCHG.
Tournament Operator:

- Tournaments are played between two solutions and the better solution is chosen and placed in the mating pool.
- Two other solutions are picked again and another slot in the mating pool is filled with the better solution.
- Each solution is made to participate in exactly two tournaments.
- The best solution in a population will win both times, thereby making two copies of it in the population.
- Any solution in a population will have zero, one, two copies in the new population.
- It has been shown (Goldberg and Deb, 1991) that the tournament selection has better or equivalent convergence and computational time complexity properties when compared to any other reproduction operator that exist in the literature.
Level_NodeSwap [LNS] Method:

- Procedure **Level_NodeSwap** *(Parent_Str, New_Str)*
  1. Randomly select a vertex \( v_i \);
  2. Select another vertex \( v_j \), such that \( v_j \) belongs to the same level of \( v_i \) (say kth level);
  3. Swap the labels of vertices \( v_i \) and \( v_j \) in *Parent_Str*;
  4. Update the level structure of *Parent_Str* after \( k^{th} \) level and call it *New_Str*;
Acceptance Criteria:

- Procedure **Acceptance Criteria** (Parent_Str, New_Str)

1. if $BW (New\_Str, G) < BW (Parent\_Str, G)$ then
2. 
3. else
4. if $BW (New\_Str, G) = BW (Parent\_Str, G)$ then
5. if $*CE (New\_Str) < CE (Parent\_Str)$ then
6. 
7. end if
8. end if
9. end if

*Critical edges* are those edges which contribute to the bandwidth. i.e. edges whose edge length $|\phi(u) - \phi(v)| = BW(\phi, G)$

*CE (str) is the number of critical edges corresponding to the labeling ‘str’.*
Algorithm 2: Simulated Annealing

1. Select cooling ratio $\alpha$ and initial temperature $t_0$
2. Select frozen temperature $t_f$, set the value of $r$
3. Select an initial solution $s_0$ by RBFS;
4. Set $t=t_0$ and count=0;
5. while $t \geq t_f$
6.   while count$\leq r$
7.     count=count+1;
8.     Obtain a neighbor $s$ by applying LNS operator on $s_0$;
9.     $\delta = BW(s,G)-BW(s_0,G)$;
10. Generate a random number $\rho$, $0 \leq \rho \leq 1$;
11. if $\rho \leq \min\{1, e^{-\delta/t}\}$ then
12.     $s_0 = s$;
13. end if
14. end while
15. count=0;
16. $t = \alpha t$;
17. end while
Computational Results:

- **Test Cases:** Harwell Boeing Sparse Matrix collection
- **Description:** Represent a large spectrum of scientific and engineering applications
- **Implementation:** MATLAB 6.5
- **Machine:** Pentium IV at 2.4 GHz
- **No. of Instances:** 18
- **Comparisons With:** Node Centroid Adjustment Method with Hill Climbing [NCHC] and Simulated Annealing [SA].
- **No of Trials:** 10 Trials for each case
**Comparative analysis of Table-1:**

- EALNS provide better solution quality.
- In most of cases EALNS attain optimum values in very short time.
- If optimum value has not been attained the results are very close to optimum values.

**Table 1: Result on Harwell-Boeing Sparse Matrix Collection**

<table>
<thead>
<tr>
<th>Graph</th>
<th>Nodes (n)</th>
<th>Edges</th>
<th>NCHC</th>
<th>SA</th>
<th>EALNS</th>
<th>Lower bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>pores1</td>
<td>30</td>
<td>103</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>ibm32</td>
<td>32</td>
<td>90</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>bcsprw01</td>
<td>39</td>
<td>46</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>bcsprw02</td>
<td>49</td>
<td>59</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>bcsstk01</td>
<td>48</td>
<td>176</td>
<td>19</td>
<td>18</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>curte54</td>
<td>54</td>
<td>124</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>will57</td>
<td>57</td>
<td>127</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>ash85</td>
<td>85</td>
<td>219</td>
<td>11</td>
<td>11</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>bcsprw03</td>
<td>118</td>
<td>179</td>
<td>15</td>
<td>19</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>nos1</td>
<td>158</td>
<td>312</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>nos4</td>
<td>100</td>
<td>247</td>
<td>14</td>
<td>11</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>Steam03</td>
<td></td>
<td>25</td>
<td></td>
<td>25</td>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>Can144</td>
<td>144</td>
<td>576</td>
<td>16</td>
<td>15</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>Can161</td>
<td>161</td>
<td>22</td>
<td></td>
<td>20</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>Dwt_59</td>
<td>59</td>
<td>163</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Dwt_66</td>
<td>66</td>
<td>193</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Dwt_72</td>
<td>72</td>
<td>147</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>
### Table-2 Computational Time (in sec):

<table>
<thead>
<tr>
<th>Graph</th>
<th>Nodes</th>
<th>Run Time NCHC (sec)</th>
<th>Run Time SA (sec)</th>
<th>Run Time EALNS (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forces_1</td>
<td>30</td>
<td>14.28</td>
<td>2.09</td>
<td>0.83</td>
</tr>
<tr>
<td>ibm32</td>
<td>32</td>
<td>497.79</td>
<td>0.32</td>
<td>30.00</td>
</tr>
<tr>
<td>bcsppwr01</td>
<td>39</td>
<td>16.00</td>
<td>0.22</td>
<td>0.77</td>
</tr>
<tr>
<td>bcsppwr02</td>
<td>49</td>
<td>72.39</td>
<td>80.14</td>
<td>6.59</td>
</tr>
<tr>
<td>bcsstk01</td>
<td>48</td>
<td>634.00</td>
<td>92.66</td>
<td>81.00</td>
</tr>
<tr>
<td>curis54</td>
<td>54</td>
<td>1.325e+003</td>
<td>10.42</td>
<td>253.00</td>
</tr>
<tr>
<td>will57</td>
<td>57</td>
<td>1.212e+003</td>
<td>276.00</td>
<td>228.00</td>
</tr>
<tr>
<td>ask85</td>
<td>85</td>
<td>425.00</td>
<td>41.20</td>
<td>253.00</td>
</tr>
<tr>
<td>bcsppwr03</td>
<td>118</td>
<td>1.9626e+003</td>
<td>122.21</td>
<td>885.00</td>
</tr>
<tr>
<td>nos1</td>
<td>158</td>
<td>375.47</td>
<td>101.70</td>
<td>19.00</td>
</tr>
<tr>
<td>nos4</td>
<td>100</td>
<td>792.85</td>
<td>462.00</td>
<td>1.87</td>
</tr>
<tr>
<td>Steam3</td>
<td>80</td>
<td>165.71</td>
<td>406.89</td>
<td>25.54</td>
</tr>
<tr>
<td>Can_144</td>
<td>144</td>
<td>3.1862e+003</td>
<td>135.39</td>
<td>97.64</td>
</tr>
<tr>
<td>Can_161</td>
<td>161</td>
<td>3.4023e+003</td>
<td>76.79</td>
<td>308.58</td>
</tr>
<tr>
<td>dwt_59</td>
<td>59</td>
<td>1.0534e+003</td>
<td>54.00</td>
<td>78.70</td>
</tr>
<tr>
<td>dwt_66</td>
<td>66</td>
<td>357.13</td>
<td>107.00</td>
<td>0.77</td>
</tr>
<tr>
<td>dwt_72</td>
<td>72</td>
<td>856.73</td>
<td>168.10</td>
<td>80.52</td>
</tr>
</tbody>
</table>

**Comparative analysis of Table-2:**

- EALNS attain the optimum solution in considerably less time.
- The CPU time is that of the best trial.
- EALNS better than NCHC in terms of solution quality as well as speed.
### Table-3 Parameter Population Size [PS]:

<table>
<thead>
<tr>
<th>Graph</th>
<th>Nodes</th>
<th>Population Size (PS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fores_1</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>ibm32</td>
<td>32</td>
<td>35</td>
</tr>
<tr>
<td>bcspwr01</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>bcspwr02</td>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>bcsstk01</td>
<td>48</td>
<td>60</td>
</tr>
<tr>
<td>curtie54</td>
<td>54</td>
<td>60</td>
</tr>
<tr>
<td>will57</td>
<td>57</td>
<td>60</td>
</tr>
<tr>
<td>ash85</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>bcspwr03</td>
<td>118</td>
<td>120</td>
</tr>
<tr>
<td>nos1</td>
<td>158</td>
<td>160</td>
</tr>
<tr>
<td>nos4</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Steam3</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Can_144</td>
<td>144</td>
<td>145</td>
</tr>
<tr>
<td>Can_161</td>
<td>161</td>
<td>161</td>
</tr>
<tr>
<td>dwt 59</td>
<td>59</td>
<td>60</td>
</tr>
<tr>
<td>dwt_66</td>
<td>66</td>
<td>70</td>
</tr>
<tr>
<td>dwt_72</td>
<td>72</td>
<td>75</td>
</tr>
</tbody>
</table>

**Parameter PS setting for EALNS:**

- Effectiveness of EALNS is dependent on population size [PS].
- PS should be at least equal to the nodes.
- If (PS-n) has larger value then the computational time also increases without any substantial improvement in the solution.
Conclusion:

- The above results show that EALNS is better than the existing approaches.
- The proposed algorithm achieved high quality solutions in considerably lesser time.
- It shows that a good local search using a knowledge based approach combined with evolutionary mechanism is very efficient in bandwidth minimization.
- With the view of future research, we expect that the notion of LNS operator given here can be further improved for the BMP.
References.


THANK YOU