ABSTRACT
The Layout Design is a ubiquitous but complex problem that requires advanced design and decision aid tools. An automated system for swiftly generating and analyzing superior layout alternatives is a much-desired approach. Various metaheuristics are often employed towards this end. However, the efficiency and applicability of such metaheuristic approaches is largely determined by the efficiency and effectiveness of module placement algorithms used. Here, one such efficient and robust placement heuristic is proposed and compared with some popular existing heuristics. Quantitative as well as qualitative fitness evaluations are used for obtaining an encompassing and realistic comparison regime. Genetic Algorithms are used to demonstrate the power of the proposed algorithm in terms of speed and quality of outcome. Notably, the proposed algorithm consistently furnishes layout alternatives carrying higher aesthetic value. The proposed algorithm is likely to support efficient and effective use of resources in layout design as well as motivate future research in related areas.
KEYWORDS
Facilities/VLSI Layout Design, Packing Problems, Metaheuristics, Placement Heuristics, BL-Algorithm

1 INTRODUCTION
Layout Design (LD) is the assembly of modules in a specified space while satisfying given subjective preferences and constraints. It is a pervasive problem in a variety of disciplines including Facilities, VLSI, and Newspaper layout designs as well as Cutting and Packing in various industries. Recently, the advent and widespread adoption of the Web and Mobile services has brought some novel, interesting, and challenging applications of intelligent and knowledge-based layout design such as E-store and E-Learning Web sites (Ahmad et al., 2004d). Intuitively, such applications differ from general LD problems in terms of highly subjective nature of preferences and objectives given a tremendously diverse target population. Nonetheless, such subjective and uncertain fitness functions, preferences, and constraints also exist in other LD domains that are traditionally treated as hard optimization problems. Instinctively, such complexity and subjectivity indicate the enormity of task in automating the LD process.

Conceivably, computer-based algorithms cannot replace human intelligence, intuition, or experience. In reality, it is irrational to expect one LD to be superior to all others for each fitness metric under consideration (Ahmad et al., 2004a; Ahmad et al., 2004b; Tompkins et al., 2002). In addition, various existing mathematical models used for layout optimization are known to be NP-Hard in strong sense (Garey and Johnson, 1979). As such, when a large number of modules are considered for layout decisions, the goal of producing an optimal or superior solution in reasonable time becomes elusive even with powerful computers. In the same league, inadequate cognitive and information processing capabilities of decision-makers often hamper the procurement of a superior outcome.

Nevertheless, computerized systems could endow valuable support to layout analysts through automated and fast generation, elaboration, articulation, enumeration, and ranking of a large number of competing Layout Alternatives (Ahmad et al., 2003; Ahmad et al., 2004b; Ahmad et al., 2004d; Akoumianakis et al., 2000; Tompkins et al., 2002). It simply implies that tackling this important and difficult problem requires intelligent and sophisticated design aids (Ahmad et al. 2004a). Recently, an intelligent and interactive decision support system for layout design has been presented that offers immense promise in this subjective problem area (Ahmad et al. 2003, Ahmad et al. 2004a). Indeed, such an automatic and knowledge-based generation and analysis of alternatives is critical to any layout planning process and its importance in terms of productivity, efficiency, and efficacy as well as quality of outcome cannot be overemphasized (Ahmad et al., 2004b; Mir and Imam, 2001; Tompkins et al., 2002).

Intuitively, various heuristic and metaheuristic techniques have significant role to play in such an automated system. Indeed, past studies have demonstrated the efficacy and promise of such heuristic and metaheuristic techniques. Among these, Genetic Algorithms are the most frequently employed and researched metaheuristics for generating superior layout alternatives. The underlying dynamics of Genetic Algorithms can efficiently generate a reasonably diverse set of superior solutions.
Such diversity and superiority of alternatives is a vital element in effective LD decisions. The core of such metaheuristic approaches is quite simple and involves treating LD as a packing problem by defining an Ordering of Modules and a Placement Heuristic for placing modules in the specified order (Ahmad et al., 2004a; Dowsland et al., 2002). Consequently, an efficient and effective placement algorithm is critical for the utility of such a synergy. Nevertheless, the existing placement heuristics lack the requisite efficiency and efficacy required in most LD work domains. As such, there is a need for developing improved heuristics. In this direction, we propose a new placement algorithm for efficiently obtaining superior layouts in both quantitative and qualitative terms. The comparison with other existing popular placement algorithms, such as the Bottom-Left or BL strategy (Jakobs, 1996) and Improved-BL or IBL (Liu and Teng, 1999) demonstrates the superiority of the new algorithm in terms of efficiency as well as quality of outcome.

The rest of the paper is organized as follows. Section 2 provides a brief overview of solution methodologies used to tackle the layout design problem. Section 3 deals with the existing and proposed placement heuristics for layout design. Section 4 outlines a Genetic Algorithm based layout design optimization approach used in comparing various placement algorithms. Section 5 provides comparison between proposed and existing placement algorithms using both quantitative and qualitative fitness appraisal regime. Section 6 concludes the paper with some future research directions.

2 LAYOUT DESIGN PROBLEM

This research draws from a wide array of domains including layout design, cutting/packing, optimization, meta-heuristics, etc. Consequently, a review of all such literature, and ensuing efforts, is beyond the scope of this paper. Nevertheless, we provide a brief overview from the automated layout design perspective for reference purposes.

2.1 Mathematical Formulations

The layout design problem has been with us since long and received considerable attention in various LD problem domains. It has been variously referred to as facilities layout, topology optimization, plant layout, machine layout, block placement, macro cell placement, layout optimization, etc. Various mathematical formulations for tackling the LD problem have been proposed in the literature. The most popular of such formulations include the quadratic assignment problem or QAP, the two-dimensional bin-packing problem or 2D-BPP, and the quadratic set-covering problem or QSC. However, the QAP does not allow control over the shape of the modules in the resulting layout and QSC requires a large number of user inputs for every module under consideration. Such laggings make these formulations somewhat incompatible for the application to general LD problems where a large number modules with fixed shape and size needs to be considered. Consequently, we use an approach similar to oriented two-dimensional finite bin packing for elaboration and comparison of existing and proposed placement heuristics.

2.2 Solution Approaches

It should be noted that even a minute shift in the location of a module in a feasible direction creates a new packing pattern without any real alteration in the topological structure (Ahmad et al., 2004b; Mir and Imam, 2001). This renders the search space to
be infinite even for very small size problems and more emphasis is given to the soft optimization methods. Various heuristic, metaheuristic, and analytical techniques have been published for tackling the LD problem. The popular heuristic techniques find solution to the problem mostly by treating it as well known quadratic assignment problem. In such an approach, the 2-dimensional plane is discretized into a grid structure. However, the QAP approach results in high computational costs arising from the discretization of modules and the packing space, thereby requiring massive parallel processing. Furthermore, it is difficult to control the shapes of modules during such a discretized approach.

As already mentioned, the existing mathematical models offer little practical advantage in dealing with problems of any genuine consequence due to the prohibitive size of associated mathematical program. Such core issues as subjectivity and vagueness of the layout design fitness evaluation objective, preferences, and constraints further exacerbate the situation. Consequently, development of fast and efficient heuristics and metaheuristics, that consistently provide ‘superior’ solutions, are the major focus in this area. Recent metaheuristics that have shown promising results include Simulated Annealing, Genetic Algorithms, Naive Evolution, and Random Search. Other solution approaches include Tree Search, Binary mixed Integer-Programming, the Network Decomposition, etc. Furthermore, some Analytical techniques that deal with continuous design space with minimal computational requirements are available (Mir and Imam, 2001). However, analytical techniques have yet to be developed to produce results comparable with advanced heuristic techniques.

3 PLACEMENT ALGORITHMS

The importance of effectively limiting the otherwise infinite search space in layout design to some reasonable and tractable subset of possible solution topologies cannot be overemphasized (Ahmad et al., 2004a; Ahmad et al., 2004b; Dowsland et al., 2002). In this regard, GA can be used in determining a sequence of modules for the use in some efficient module placement strategies. Such placement algorithms determine the position of modules in the resulting packing pattern. The position of modules in the layout is used for calculating the fitness of the sequence of modules generated by GA. A module placement algorithm takes one gene (or module) at a time from a chromosome (or the sequence of modules) and determines its position in the packing space based on pre-specified steps. Indeed, it has been argued that computational cost of such a GA based layout optimization process is determined by the cost of the module placement algorithm (Dowsland et al., 2002; Jakobs, 1996). Consequently, an efficient module placement strategy that generates superior quality layouts is critical for the efficacy of such an endeavor (Ahmad et al., 2004a).

3.1 BL-Algorithm

Recently, the BL placement algorithm has drawn considerable attention from researchers (Jakobs, 1996; Liu and Teng, 1999; Dowsland et al., 2002). It calls for placing a module at the bottom-most and left-most feasible position. The process is illustrated in Figure 1. The apparent advantages of such approaches include speed and simplicity (Dowsland et al., 2002).

Following are the main steps involved in the simple BL algorithm:
Let $Blocks = \text{No. of Modules at hand for Placement.}$

1) Place module $M_1$ at the bottom-left corner of the bin

2) For $K = 2$ to $Blocks$
   - Shift module $M_K$ alternately, beginning from upper-right corner of packing space, as far as possible to the bottom and then as far as possible to the left.

Next $K$

3) Stop if no room for more modules.

![Figure 1: Poor Space Utilization with BL.](image1)

![Figure 2: Optimal Packing Not Possible with BL.](image2)

### 3.2 Deficiencies of BL-Algorithm

The popularity of the BL is derived from simplicity of the underlying notion, ease of implementation, and speed of execution. However, BL has such inherent shortcomings as poor space utilization and inability to obtain some simple optimal solutions. For instance, if we know an optimal packing pattern of $n$ modules that even fulfills the BL-condition, we cannot always write out a permutation for the BL-algorithm corresponding to it. Simply stated, the optimal packing pattern cannot be obtained by the BL-algorithm even if all permutations are enumerated as the case in example shown in Figure 2. In addition, BL is not very effective in incorporating such qualitative design consideration as symmetry, balance, equilibrium, coherence, homogeneity, etc.

Furthermore, the BL is more suitable for cases where the objective is geared towards minimizing the height of the packing pattern (for instance, stamping patterns out of a fixed-width rectangular piece of steel sheet). However, if two packing patterns have the same height or consume equal amount of space then their fitness values are same. Nevertheless, one of the packing patterns can be deemed 'superior' to the other based on other objectives such as symmetry. In addition, the BL algorithm converges modules at the bottom-left corner of the packing space that might not be a useful strategy in many cases. For instance, some facility layout design might require modules to be placed around some focal point(s).

### 3.3 Improved-BL

Various improvement plans have been proposed for the BL such as Improved-BL or IBL (Liu and Teng, 1999). Such improved strategies consist of refinement in placement decisions in the Step 2 of BL by giving priority to a shift towards bottom and some allowance for module rotation. However, even these improved algorithms encounter such problems as lack of aesthetic value and dead-area surrounded by
placed modules that could not be utilized, as shown in Figure 1. Nevertheless, such improvement schemes are quite popular. Consequently, we have included IBL in our comparison analyses. However, our implementation of IBL does not involve rotation of modules in accordance with our intended work domain that involves only oriented modules.

3.4 Proposed Algorithm

The proposed module placement algorithm is motivated by the fact that for any given packing space the number of modules available for placement is a small integer. In addition, if we restrict our packing decisions only to the corners of the ‘in-place’ modules then for a given module there are only $O(n)$ possible locations. Consequently, the combinatorial intractability should not deter the use of some fast pseudo-exhaustive search, which aims at improving the space utilization as well as layout quality, in a hierarchical manner (Ahmad et al. 2004a).

Here we outline the proposed module placement algorithm called Minimization of Enclosing Rectangle Area (MERA) algorithm. The name implies the underlying notion where a reduction of the rectangular area of the packing pattern is sought during all placement decisions. The refinement part in the placement algorithm is a simple pseudo-exhaustive search. This pseudo-exhaustive placement technique is summarized in the following steps:

Let:
- $Blocks = \text{No. of Modules at hand for Placement}$
- $Nplaced = \text{No. of Modules Already Placed}$
- $y_i = y$-coordinate of the bottom-left corner of the $i^{th}$ module.
- $newOBJ = \text{Area of the Enclosing Rectangle} + \frac{(y_i + \text{width of Enclosing Rectangle})}{2}$

1) Place module 1 at the bottom-left corner of the page
2) Set $OBJ$ to a big value
3) FOR $K = 2$ to $Blocks$
   FOR $L = 1$ to $NPlaced$
   FOR $A = 1$ to 4
   FOR $B = 1$ to 4
   Place corner $B$ of $M_K$ on corner $A$ of $M_L$
   Check Overlap conditions
   Check Boundary conditions
   IF both conditions satisfied THEN
   Calculate the $newOBJ$
   IF $newOBJ$ is less than $OBJ$ THEN
   $OBJ = newOBJ$
   Save placement of module $M_K$
   ENDIF
   ENDIF
   END $B$
   END $A$
   END $L$
   END $K$
4) Stop if no room for more modules.

In the above pseudo-code, index $A$ represents four corners of an in-place module ($M_L$) and index $B$ represents four corners of an in-coming module ($M_K$). The Step 2 proceeds by exploring the placement opportunities for each of the four corners of an in-coming
module at each of the four corners of an in-place module. The second term in newOBJ is meant to bias placement to the bottom of packing, a general packing preference in bin-packing context.

The computational cost of BL-algorithm is $O(n^2)$ as each in-coming module can be shifted a maximum of $i$ times as each shift is limited by one of the $i-1$ already placed modules or by the boundaries of the bin. However, in case of MERA, each in-coming module can be placed at a maximum of $16i$ corner points (a very loose bound) where $i-1$ modules are already in place. As such, theoretically, MERA also has the same $O(n^2)$ computational cost as the case for BL (Jakobs, 1996; Liu and Teng, 1999). Furthermore, such a small increase in computational complexity is deemed quite reasonable and tolerable, as demonstrated by the results in Section 5.

4 GENETIC ALGORITHM

We used Genetic Algorithms for the optimization part of our studies. Indeed, past studies have shown that the GA based metaheuristic solution approach for unequal area facilities layout outperforms other available heuristics and metaheuristics by large margins. The encoding scheme employed for the GA represents a layout using a sequence of modules. For example: \{12, 4, 9, 20, 11, 14, 2, 16, 13, 15, 1, 3, 18, 10, 17, 5, 19, 7, 6, 8\} shows a sequence of 20 modules to be placed in a given packing space. A pre-specified and static population size of 25 is used in the evolution process. The initialization step involved random generation of 25 sequences of modules ($S_1, S_2, \ldots S_{25}$).

Various genetic operators selected for the GA are outlined here. The selection operator selects individual layouts for genetic operations. Intuitively, we selected the biased Roulette selection scheme as it gives more weightage to fitter solutions.

The mutation operator uses a single solution to generate new individuals. In our context, mutation results in minor changes, such as swapping of two modules or deletion/insertion of a module, in an existing layout. The mutation rate is selected to be high for reasons discussed later. The mutation operators used in our experiments involved exchanging elements of two randomly selected layout subsequences and addition, deletion, or swapping of genes.

The crossover operator creates one or more offspring solutions by combining two or more parents. In our context, crossover results in combining segments of two existing layouts in order to generate a new layout. The crossover operator we used involves two parents (say $S_j$ and $S_k$) selected randomly based on their ranks in the population. Crossover consists of following steps. Copy $q$ elements of the sequence $S_j$ at a random position $p$ in the new sequence $S_{new}$. Fill up the remaining elements of $S_{new}$ with other elements of $S_k$ in the same order. Our desire for an adequate exploitation of good intermediate solutions in our search process resulted in the selection of a low crossover rate of 20%.

Keeping in view of the cost of placement algorithms, we opted for a replacement strategy in which the GA sorts out the worst individual after a new offspring layout is created on an ongoing basis. As a result, ‘superior’ offsprings could influence the layout solution quality. However, such strategy pronounces the need for high mutation rate to ensure population diversity. After some preliminary test, we opted to terminate the GA past 1000 iterations, a seemingly low but practically sufficient endeavor to achieve results very close to those generated by $50,000^+$ iterations.
The most challenging and application specific task in any particular problem domain exploiting GA is the definition of the fitness function. A GA uses fitness function to differentiate ‘superior’ and ‘inferior’ layout solutions and tends to converge to solutions superior against it. The ideal, though somewhat impractical, course would be to let layout designers determine the fitness through visual evaluation. However, in quest of a more encompassing comparison regime, we used the minimization of Packing Height (H) coupled with relatively new metrics like the minimization of Module Tightness (MT) and the maximization of Contiguous Remainder (CR). All these metrics provide some clue towards the degree of space utilization (Ahmad et al., 2004a; Ahmad et al., 2004b; Jakobs 1996, Liu and Teng, 1999, Hopper and Turton, 2001). Nevertheless, such metrics fail to appraise such qualitative aspects as symmetry of the layout.

Packing height is by far the most commonly used yardstick in comparative analyses available in bin-packing related literature (Ahmad et al., 2004c). However, the CR is a relatively new metric. Such rigid and myopic metrics have their laggings that can be elaborated by Figure 3. It can be seen that the two packing patterns have same height; nevertheless, the pattern B is obviously more desirable than pattern A in a bin-packing context. In this regard, CR, seems to be a better choice than the other two. It is the area of the largest contiguous unused portion of the bin still available for further packing. For instance, the contiguous remainder of pattern A Figure 3 is less than that of B. However, contiguous remainder is also a quite rigid fitness measure with its own laggings. For instance, if there is a long and narrow strip of space available where no module can be placed, the common sense does not support assigning it a high fitness value than a layout that has smaller CR but sufficient to place one or more modules.

Figure 3: Two Packing Patterns with same Height

5 RESULTS AND DISCUSSION

A computer program is written in Visual BASIC to implement the BL, IBL, and MERA algorithms as well as the proposed GA based optimization component. In our studies, all aspects are kept identical except the placement algorithm. Consequently, these studies do not account for any interactions between the placement algorithm and some GA parameters. The resultant computer program is used on Intel Xeon 3.06 GHz processor and 256MB of RAM under Windows XP. The comparative analyses are shown in Tables 1 and 2.

The comparative analyses are based on quantitative metrics such as Packing Height, Module Tightness, and Contiguous Remainder. Various fitness metrics that can capture some sense qualitative aspects such as symmetry, coherence, balance, etc. are available in literature (Ahmad et al., 2004b; Ngo, 2001). However, such metrics
cannot replace human intuition and aesthetic perception. Consequently, we requested a couple of experienced researchers in the layout design area to evaluate and rank the qualitative fitness, with regard to the layout symmetry, on a scale of 1 to 10. We randomly generated 50- and 100-module problems for comparison purposes. The performance of the BL, IBL, and MERA placement strategies for 100 random sequences each problem instance are presented in Table 1.

<table>
<thead>
<tr>
<th>No. of Modules</th>
<th>Objective</th>
<th>Tech.</th>
<th>Wins</th>
<th>Best</th>
<th>Worst</th>
<th>Mean</th>
<th>Std.Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>Height (Ideal = 55)</td>
<td>BL</td>
<td>1</td>
<td>66</td>
<td>99</td>
<td>78.9</td>
<td>5.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL</td>
<td>3</td>
<td>68</td>
<td>91</td>
<td>77.2</td>
<td>4.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA</td>
<td>96</td>
<td>64</td>
<td>77</td>
<td>69.4</td>
<td>3.11</td>
</tr>
<tr>
<td></td>
<td>Module Tightness (Ideal = 100%)</td>
<td>BL</td>
<td>1</td>
<td>83.49</td>
<td>56.22</td>
<td>70.21</td>
<td>4.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL</td>
<td>2</td>
<td>81.03</td>
<td>60.54</td>
<td>71.72</td>
<td>4.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA</td>
<td>97</td>
<td>85.63</td>
<td>72.28</td>
<td>80.38</td>
<td>3.49</td>
</tr>
<tr>
<td></td>
<td>Contig. Remainder (Ideal = 4500)</td>
<td>BL</td>
<td>0</td>
<td>3848.0</td>
<td>2365</td>
<td>3404.6</td>
<td>248.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL</td>
<td>2</td>
<td>4025.0</td>
<td>2976</td>
<td>3600.6</td>
<td>191.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA</td>
<td>98</td>
<td>4310.2</td>
<td>3939</td>
<td>4049.4</td>
<td>89.7</td>
</tr>
<tr>
<td></td>
<td>Quality (Ideal = 10)</td>
<td>BL</td>
<td>0</td>
<td>4.0</td>
<td>1.5</td>
<td>2.95</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL</td>
<td>1</td>
<td>4.5</td>
<td>1.0</td>
<td>3.25</td>
<td>1.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA</td>
<td>9</td>
<td>7</td>
<td>3.5</td>
<td>4.9</td>
<td>1.15</td>
</tr>
<tr>
<td>100</td>
<td>Height (Ideal = 100)</td>
<td>BL</td>
<td>0</td>
<td>126.4</td>
<td>149.9</td>
<td>141.2</td>
<td>4.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL</td>
<td>1</td>
<td>125.3</td>
<td>149.7</td>
<td>138.4</td>
<td>5.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA</td>
<td>99</td>
<td>113.5</td>
<td>139.5</td>
<td>121.1</td>
<td>5.28</td>
</tr>
<tr>
<td></td>
<td>Module Tightness (Ideal = 100%)</td>
<td>BL</td>
<td>0</td>
<td>78.4</td>
<td>66.1</td>
<td>70.9</td>
<td>2.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL</td>
<td>0</td>
<td>79.0</td>
<td>66.2</td>
<td>71.8</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA</td>
<td>100</td>
<td>88.3</td>
<td>69.8</td>
<td>83.3</td>
<td>3.26</td>
</tr>
<tr>
<td></td>
<td>Contig. Remainder (Ideal = 5000)</td>
<td>BL</td>
<td>0</td>
<td>3117.8</td>
<td>879.3</td>
<td>1995.7</td>
<td>442.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL</td>
<td>0</td>
<td>3147.3</td>
<td>1591.7</td>
<td>2491.9</td>
<td>361.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA</td>
<td>100</td>
<td>4381.1</td>
<td>3694.2</td>
<td>4100.6</td>
<td>141.3</td>
</tr>
<tr>
<td></td>
<td>Quality (Ideal = 10)</td>
<td>BL</td>
<td>0</td>
<td>4</td>
<td>1.5</td>
<td>2.65</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL</td>
<td>0</td>
<td>3.5</td>
<td>2.0</td>
<td>2.75</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA</td>
<td>10</td>
<td>6</td>
<td>2.5</td>
<td>4.05</td>
<td>1.17</td>
</tr>
</tbody>
</table>

Table 1: Comparison of BL, IBL, and MERA for 100 random sequences.

The average CPU time taken by BL and MERA for 100 random sequences of 100-module problem is shown in Figure 4. Apparently, the average time taken by MERA increases drastically with the problem size. However, this plot is quite deceiving from various practical standpoints, explained as follows. Our experience has demonstrated that a very few random module sequences almost always, if not always, furnish at least one layout with MERA-alone that is superior to the best obtained after a whole cycle of BL+GA or IBL+GA. It is evident from Figure 6 as well as from contrasting the results shown in Table 1 with those in Table 2. It happened in more than 75% random sequences of 50-module problem and in more than 85% random sequences of the 100-module problem during repeated experiments. Furthermore, a relative comparison of average CPU time taken by the two algorithms (MERA/BL) shows that an increase in
the problem size only results in a linear increase in relative computational cost, as illustrated in Figure 5.

**Figure 4**: Average Time Elapsed per 100 iterations for MERA and BL

**Figure 5**: Ratio of Average Time Elapsed per 100 iterations for MERA and BL

In addition, the resulting superior quality and diversity of layout alternatives obtained through MERA sanction a relatively higher computational cost a worthwhile trade-off. Moreover, the performance of BL and IBL is known to deteriorate dramatically with the increase in the problem size as can be seen from Tables 1 and 2, and as demonstrated by a series of previous studies (Ahmad et al. 2004c; Jakobs 1996; Liu and Teng 1999; Hopper and Turton 2001). In contrast, MERA results in significantly higher performance improvements for larger problems furnishing another cogent incentive for resorting to a computationally costly approach.

The average of 10 runs of GA using population size of 25, mutation rate of 0.8, and crossover rate of 0.2 is shown for BL, IBL, and MERA in Table 2. It can be seen that MERA results in significant improvements in the layout fitness, especially for larger problems.

<table>
<thead>
<tr>
<th>No. of Modules</th>
<th>Objective</th>
<th>Tech.</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>Contig. Remainder (Ideal = 4500) Higher the Better</td>
<td>BL+GA</td>
<td>4111 (91.3%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL+GA</td>
<td>4232 (94.0%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA+GA</td>
<td>4345 (96.5%)</td>
</tr>
<tr>
<td></td>
<td>Quality (Ideal = 10) Higher the Better</td>
<td>BL+GA</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL+GA</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA+GA</td>
<td>4.5</td>
</tr>
<tr>
<td>100</td>
<td>Contig. Remainder (Ideal = 5000) Higher the Better</td>
<td>BL+GA</td>
<td>3432 (68.6%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL+GA</td>
<td>3905 (78.1%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA+GA</td>
<td>4709 (94.2%)</td>
</tr>
<tr>
<td></td>
<td>Quality (Ideal = 10) Higher the Better</td>
<td>BL+GA</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBL+GA</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERA+GA</td>
<td>4.0</td>
</tr>
</tbody>
</table>

**Table 2**: Comparison of BL, IBL, and MERA based on optimization using GA.

Furthermore, as depicted in Figure 6, even the first few iterations with MERA produces results comparable, or even superior, to the whole GA cycle of BL or IBL involving a large number of iterations. In contrast, BL and IBL start with inferior solutions and converge to comparatively better solutions against the given fitness...
measure. In short, MERA not only results in better quality alternatives but also provides those faster than those obtained with BL or IBL. It demonstrates that the computational cost of MERA is not prohibitive in very efficiently achieving results better than BL or IBL. Furthermore, it can be seen from Table 2 that GA provides large improvements with MERA in comparison to BL or IBL. These improvements are dramatically large for larger problems.

Briefly, MERA results in a more efficient, effective, and superior layout optimization than other existing algorithms. It demonstrates that MERA, somehow, captures the dynamics of the LD problem more aptly. Furthermore, MERA is quite simple to understand and implement just as BL and IBL. Consequently, BL and IBL do not hold any superiority over MERA in terms speed, quality, ease of understanding or ease of implementation.

**Figure 6**: Convergence of GA with BL, IBL, and MERA for 50-module problem

6 CONCLUSION

The layout design is a difficult and frequently encountered problem in various work domains. In this paper, a new efficient, effective, and robust module placement strategy has been proposed for obtaining superior layout alternatives. Studies with such metaheuristics as Genetic Algorithms demonstrate the power of the new placement strategy to outperform some popular existing strategies in speed as well as quality of outcome. It is particularly suited for applications where aesthetic values of the layout bear importance. Such strengths make it quite pertinent to the development of some automated knowledge-based decision support tool for layout design. We believe that this research would result in increased efficiency and productivity of layout designers as well as facilitation of future research in related areas.

Acknowledgements:

Authors are grateful to Dr. F. Karray of Systems Design Engineering, University of Waterloo, for his guidance and support during this on going research project. Special thanks to an anonymous technical support specialist at VIP-PlanOpt™
(www.PlanOpt.com) for valuable electronic discourse prompting a better understanding of the dynamics of this intricate problem.

REFERENCES