Adaptive Bias Simulated Evolution Algorithm for Placement

Sadiq M. Sait, Habib Youssef, Hussain Ali
Contents

- Introduction
  - VLSI Placement
  - Simulated Evolution Algorithm
  - Bias Schemes
- Proposed Adaptive Selection Bias
- Experiments and Results
- Conclusion
VLSI Placement

- It consists of arranging circuit blocks on a layout surface such that (multiple) cost objectives are optimized.

Simulated Evolution (SE) Algorithm

» SE is a general meta-heuristic proposed by Kling and Bannerjee (1987).

» Useful for the solution of combinatorial optimization problems

» It emphasizes the behavioral link between parent and offspring, or between reproductive populations, rather than the genetic link
SE Algorithm

- Algorithm starts with a valid initial solution
- Goodness for each element of the current solution is computed
- Using goodness, elements are selected probabilistically. A fixed Bias value is used.
- Constructive perturbation of selected elements
- If objective(s) are not satisfied then continue
Simulated Evolution($B$, $\Phi_{initial}$, Stopping Condition)

\begin{algorithm}
\textbf{Input:}
\begin{itemize}
  \item $B$ = Problem data.
  \item $\Phi_{initial}$ = Initial solution.
  \item Stopping Condition = Stop condition.
\end{itemize}
\textbf{Output:}
\begin{itemize}
  \item $\Phi$ = Complete Solution.
\end{itemize}

$e_i$ = Individual cell in $\Phi$.
$O_i$ = Lower bound on cost of $i^{th}$ cell.
$C_i$ = Current cost of $i^{th}$ cell in $\Phi$.
$g_i$ = Goodness of $i^{th}$ cell in $\Phi$.
$S$ = Queue to store the selected cells.
$ALLOCATE(e_i, \Phi_i)$ = Function to allocate $e_i$ in partial solution $\Phi_i$.

\textbf{Repeat}

\textbf{EVALUATION:} ForEach $e_i \in \Phi$ DO
\begin{algorithmic}
\State $g_i = \frac{O_i}{C_i}$
\end{algorithmic}
end

\textbf{SELECTION:} ForEach $e_i \in \Phi$ DO
\begin{algorithmic}
\State \textbf{IF} Random > Min($g_i + B$, 1) \textbf{THEN}
\begin{algorithmic}
\State $S = S \cup e_i$; Remove $e_i$ from $\Phi$.
\end{algorithmic}
\end{algorithmic}
end
\begin{algorithmic}
\State Sort the elements of $S$
\end{algorithmic}

\textbf{ALLOCATION:} ForEach $e_i \in S$ DO
\begin{algorithmic}
\State $ALLOCATE(e_i, \Phi_i)$
\end{algorithmic}
end
\textbf{Until} Stopping Condition is satisfied
\textbf{Return} Best solution.
\textbf{End} (Simulated Evolution)
Goodness of individual cell $C_i$ which is a part of nets \{ $V_1$, $V_2$, $\ldots$, $V_k$ \} is computed as follows

$$g_{ci} = \frac{1}{k} \sum_{j=1}^{k} \min \left( \frac{L_{vj}^*}{L_{vj}}, 1.0 \right)$$

where $L_{vj}^*$ and $L_{vj}$ are respectively optimum and actual wire length of net $V_j$. 
Selection (SE)

» A cell \( C_i \) is selected if the following condition is satisfied:

\[
g_{ci} + BIAS < \text{Random}[0,1]
\]

» where BIAS is a user-defined parameter

» Selected cells are removed from the solution.
The selected cells are placed on the layout using "sorted individual best fit (SIBF)" scheme.

In SIBF each selected cell is placed on a location which results in maximum reduction in cost. That location is marked as occupied.

However, for multiple and conflicting objectives identifying such a location requires tradeoffs.
The Need for Bias

- The accurate computation of goodness is not possible as it requires the knowledge of optimum cost.
- The bias is used to inflate or deflate the goodness of elements, thus controls the size of selection set.
The Need for Bias

- Lower bias values lead to higher execution time and quality of solution is also degraded due to un-certainty created by large perturbations.
- A high bias value results in small selection set and the quality of solution is poor due to limitations of algorithm to escape local minima.
Solutions for finding suitable Bias Value

- **Fixed Bias (Kling and Banerjee)**
  - Make several trial runs with different bias values
  - **Disadvantages**
    - Not a general bias value
    - Trial runs require excessive execution time
Solutions for finding suitable Bias Value

- **Normalized Goodness**
  - Normalize individual goodness values and use a zero bias value

\[
g'_i = 0.05 + 0.95 \times \frac{g_i - g_i^{\min}}{g_i^{\max} - g_i^{\min}}
\]

- Normalized individual goodness in the range 0.05 and 0.95.
We propose automatic estimation of Bias parameter by the algorithm as a function of current solution quality.

At $K^{th}$ iteration bias $B_k$ is computed as

$$B_k = 1 - G_k$$

Where $G_k = \text{average goodness of all the elements}$
Proposed Adaptive Selection Bias

**Features**

- Bias is not arbitrarily selected and no trial runs are required. Adaptive bias automatically adjusts according to the problem state.
- It controls the size of the selection set.
  - Poor Quality Solution $\Rightarrow$ High Bias $\Rightarrow$ Controlled Selection Set $\Rightarrow$ saves the algorithm from large perturbations
  - High Quality Solution $\Rightarrow$ Low Bias $\Rightarrow$ Sufficient Selection Set $\Rightarrow$ protection against early convergence of algorithm
Proposed Adaptive Selection Bias

- Features

  - At iteration $k$ only elements with $\text{goodness} < G_k$ have non zero probability of selection. Hence search is focused on relocating poorly placed elements.
Experiments and Results

» We compare fixed bias, normalized goodness and adaptive bias on VLSI cell placement problem.

» The tests are carried out on ISCAS-85 benchmarks.

» The comparison is based on quality of solution and execution time.
Experiments and Results

Comparison of solution quality normalized in the range (0 ÷ 1) and algorithm execution time (in minutes).

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Fixed Bias</th>
<th>N.Goodness</th>
<th>A. Bias</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q</td>
<td>T</td>
<td>Q</td>
</tr>
<tr>
<td>fract</td>
<td>0.65</td>
<td>15.0</td>
<td>0.64</td>
</tr>
<tr>
<td>c499</td>
<td>0.64</td>
<td>17.5</td>
<td>0.64</td>
</tr>
<tr>
<td>c532</td>
<td>0.67</td>
<td>52.0</td>
<td>0.69</td>
</tr>
<tr>
<td>c880</td>
<td>0.77</td>
<td>62.0</td>
<td>0.75</td>
</tr>
<tr>
<td>c1355</td>
<td>0.74</td>
<td>372.0</td>
<td>0.65</td>
</tr>
<tr>
<td>struct</td>
<td>0.76</td>
<td>1780.0</td>
<td>0.67</td>
</tr>
<tr>
<td>c3540</td>
<td>0.75</td>
<td>1800.0</td>
<td>0.60</td>
</tr>
</tbody>
</table>

Click Here to upgrade to Unlimited Pages and Expanded Features
Experiments and Results

Comparisons

1. Quality of Solution vs. iterations
2. Execution Time
3. Cardinality of Selection set against frequency of solutions
4. Average goodness of the selected cells
5. Quality of solution against iterations of algorithm for different quality ranges.
Experiments and Results

Quality of Solution for fixed bias, adaptive bias and normalized goodness SE algorithm for placement

Circuit: c1355

Solution Quality normalized in the range (1,0)

Iterations

- Fixed Bias
- Adaptive Bias
- Normalized Goodness
Experiments and Results

Execution time for best fixed bias, variable bias and normalized goodness SE algorithms

Circuit: c1355

<table>
<thead>
<tr>
<th>Execution time in minutes</th>
<th>Best Fixed Bias</th>
<th>Variable Bias</th>
<th>Normalized Goodness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>100</td>
<td>600</td>
</tr>
</tbody>
</table>
Experiments and Results

Cardinality of Selection Set for fixed bias, variable bias and Normalized Goodness SE algorithms vs. frequency of generations

Circuit: c1355

- 0.2
- Variable Goodness (1-G)
- Normalized Goodness
Experiments and Results

Average Goodness of Selected Cells vs. Iterations

Circuit: c1355

- Fixed Bias
- Adaptive Bias
- Normalized Goodness
Experiments and Results

5 Quality of solution against iterations of algorithm for different quality ranges.

![Graphs showing quality of solution vs. iterations for different quality ranges.]

- Left: Fixed Bias Simulated Evolution Algorithm
- Right: Adaptive Bias Simulated Evolution Algorithm

- Y-axis represents the number of solutions.
- X-axis represents iterations.
- Different quality ranges are indicated by distinct line styles and colors.
Conclusion

» Adaptive Bias scheme where bias value automatically evolves as the search progresses.

» SE algorithm becomes more adaptable to the overall quality of solution.

» Bias is no longer an algorithm parameter that must be tuned for every problem instance.