ADAPTIVE BIAS SIMULATED EVOLUTION ALGORITHM FOR PLACEMENT

Habib Youssef *Sadiq M. Sait* *Hussain Ali*

Computer Engineering Department
King Fahd University of Petroleum and Minerals
KFUPM Box # 673, Dhahran 31261, Saudi Arabia
e-mail: {youssef,sadiq,hussain}@ccse.kfupm.edu.sa

ABSTRACT

Simulated Evolution (SE) is a general meta-heuristic for combinatorial optimization problems. A new solution is evolved from current solution by relocating some of the solution elements. Elements with lower goodnesses have higher probabilities of getting selected for perturbation. Because it is not possible to accurately estimate the goodness of individual elements, SE resorts to a *Selection Bias* parameter. This parameter has major impact on the algorithm run-time and the quality of the solution subspace searched. In this work, we propose an adaptive bias scheme which adjusts automatically to the quality of solution and makes the algorithm independent of the problem class or instance, as well as any user defined value. Experimental results on benchmark tests show major speedup while maintaining similar solution quality.

1. INTRODUCTION

Kling and Banerjee proposed Simulated Evolution (SE) as a general meta-heuristic [1] for the solution of combinatorial optimization problems. Starting from a given initial solution, SE repetitively executes the following steps until stopping conditions are met: **Evaluation**, **Selection**, and **Allocation** (see Figure 1). The **evaluation** step estimates the **goodness** $g_i \in (0, 1)$ of each element i in its current location. The goodness of an element is a ratio of its optimum (minimum) cost O_i to current cost estimate C_i. A goodness near 1 indicates a highly fit individual. In **selection** step, the algorithm probabilistically selects unfit elements. Elements with low goodness values have higher probabilities of getting selected for relocation. The selected elements are removed from the solution and reassigned one at a time to new locations in a constructive **allocation** step. The objective of this step is to improve their goodness values, thereby reducing the overall cost of the solution.

The accurate computation of goodness in SE is not possible as it requires the knowledge of optimum cost.

Algorithm *Simulated_Evolution*($B, \Phi_{initial}, StoppingCondition$)
NOTATION
B= Bias Value. Φ= Complete Solution.
e_i= Individual cell in Φ. O_i= Lower bound on cost of i^{th} cell.
C_i= Current cost of i^{th} cell in Φ. g_i= Goodness of i^{th} cell in Φ.
S= Queue to store the selected cells.
$ALLOCATE(e_i, \Phi_i)$=Function to allocate e_i in partial solution Φ_i
Repeat

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

SELECTION: **ForEach** $e_i \in \Phi$ **DO**
begin
IF *Random* $> Min(g_i + B, 1)$
THEN **begin**
$S = S \bigcup e_i$: Remove e_i from Φ.
end
end

Sort the elements of S

ALLOCATION: **ForEach** $e_i \in S$ **DO**
begin
$ALLOCATE(e_i, \Phi_i)$
end

Until *Stopping Condition is satisfied*
Return Best solution.
End (*Simulated_Evolution*)

Figure 1: Simulated Evolution algorithm.

In order to compensate errors made in the estimation of goodness, Kling and Banerjee proposed the use of **fixed selection bias** to inflate or deflate the goodness of elements. A high positive value of bias decreases the probability of selection of elements thus reducing the size of selection set. Lower bias values lead to higher execution times caused by larger selection sets. The quality of solution is also degraded due to uncertainty created by large perturbations [1]. Similarly, for high bias values the size of the selection set is small, and the quality of solution is poor due to limitations of the algorithm to escape local minima [1].

The approach recommended by Kling and Banerjee for finding a suitable bias value for a given problem instance is to make several trial runs with different bias values. However, these trial runs will result in finding the best bias value for only that instance of the problem. Therefore, it can't be used as a general value for any set of problems. Furthermore, trial runs require excessive execution time. Other researchers have pro-

0-7803-6685-9/01/$10.00©2001 IEEE

posed a **normalized goodness** scheme which consists of a linear transformation that spreads cell goodnesses on the entire interval (0,1), thus avoiding the use of bias altogether [2, 3, 4]. However, as we will see, even this scheme fails as the size of the problem increases. It takes excessive execution time and deteriorates quality of the best solution found.

This work proposes **adaptive bias** approach where the bias value is adjusted as the solution is evolved. Experiments show that the proposed bias scheme results in similar quality solutions as obtained with fixed bias, with major savings in execution times. In this work, the test problem used is cell placement problem of VLSI design which is a generalization of quadratic assignment problem [5]. Placement can be formulated as follows [6]: *Given a finite set E of distinct movable cells and a finite set L of locations, a state is defined as an assignment function $S : E \rightarrow L$ satisfying certain constraints.* The objective of placement is to assign each cell $e_i \in E$ to a unique location L_j such that some criteria are optimized [6]. Generally minimization of wire-length has been widely used as the objective of VLSI placement.

2. NORMALIZED GOODNESS

One solution to the problem of finding suitable bias value in advance is to normalize individual cell goodnesses [2, 3, 4] and use zero bias value. The objective of the normalization is to spread the goodness within the range (*Upper*, *Lower*) where $Upper + Lower \leq 1.0$. The advantage of this approach is that the SE algorithm becomes independent of bias value. Yuh et. al. [2] have normalized individual *score* in the range 0.05 and 0.95. Using this normalization method, individual cell goodnesses are computed as follows:

$$g_i' = 0.05 + 0.95 \times \frac{g_i - g_i^{\min}}{g_i^{\max} - g_i^{\min}} \qquad (1)$$

where g_i is the actual goodness of cell i, $g_i^{\min} = \min_i(g_i)$, $g_i^{\max} = \max_i(g_i)$, and g_i' is the goodness of cell i after normalization. Using Equation 1, the best cell in a solution will have a goodness of 0.95, and the worst cell will have a goodness of 0.05.

3. PROPOSED ADAPTIVE SELECTION BIAS

This work proposes an **adaptive bias** scheme where the bias parameter is automatically estimated by the algorithm as a function of current *solution quality*. Bias value evolves with the search process. As the overall goodness of the solution improves, bias is proportionally adapted. Overall goodness of a solution is measured by the average across all cells in the solution. It is an estimate of how near is each cell to its optimum position. At k^{th} iteration bias B_k is computed as follows.

$$B_k = 1 - G_k \qquad (2)$$

where G_k is the average goodness of all the cells at the end of k^{th} iteration. This approach presents several advantages.

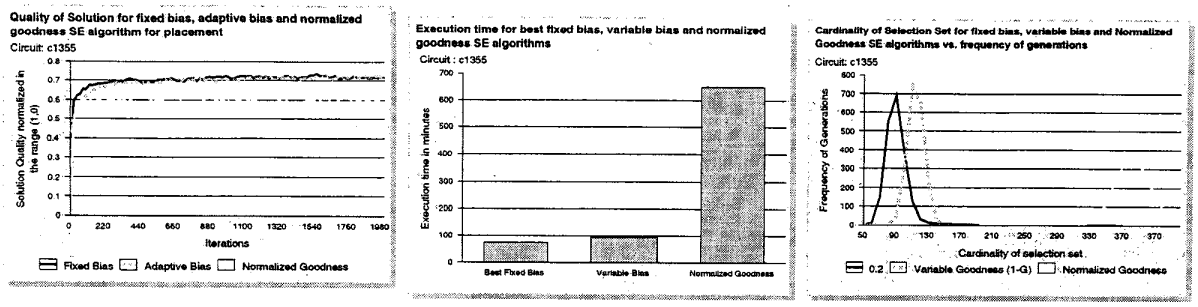
1. Bias value is not arbitrarily selected and no trial runs are required to find the suitable bias value. The adaptive bias automatically adjusts according to the problem state.
2. The state space search still corresponds to an ergodic Markov chain, and therefore convergence properties of SE algorithm are maintained[1].
3. For poor quality solutions, the average goodness is low, resulting in a high bias value. This will make sure that the size of selection set is not excessively large. It will save the algorithm from making excessively large perturbations.
4. For good quality solutions, the average goodness is high, leading to a low bias value. This will result in the selection of a sufficient number of cells which will protect the algorithm from early convergence.
5. At iteration k only elements with $goodness < G_k$ have non zero probability of getting selected for perturbation. Hence the search is always focused on relocating poorly placed elements.

4. RESULTS AND DISCUSSION

In this section we compare three selection bias strategies namely **fixed bias** (the original SE scheme proposed by Kling and Banerjee), **normalized goodness** with zero bias, and our proposed **adaptive bias**. The test problem is VLSI cell placement. The tests are carried out on ISCAS-85 benchmark circuits. Initial solutions are randomly generated and algorithms are executed for a fixed number of iterations on SUN Ultra Sparc-1 Workstations. This comparison is based on the quality of the solution and algorithm execution time. The quality of a solution S, $Q(S) = 1 - NC(S)$ where $NC(S) \in (0, 1)$ is the normalized cost of that solution.

Table 1 compares the quality of final solution and execution time of SE with different bias schemes. The

[1]Proof left out due to lack of space.



(a) (b) (c)

Figure 2: Comparison of different bias schemes: (a) quality of solution; (b) execution time; and (c) cardinality of selection set against frequency of solutions for different bias schemes for the SE algorithm.

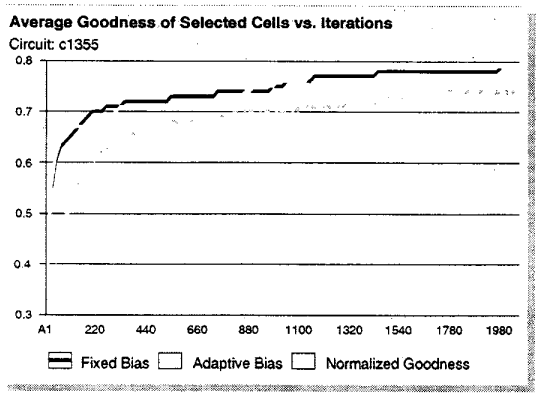


Figure 3: Comparison of different bias schemes depicting average goodness of the selected elements.

Circuit	Fixed Bias		N.Goodness		A. Bias	
	Q	T	Q	T	Q	T
fract	0.65	15.0	0.64	5.4	0.64	1.0
c499	0.64	17.5	0.64	20.4	0.63	2.8
c532	0.67	52.0	0.69	24.4	0.65	5.6
c880	0.77	62.0	0.75	72.0	0.77	17.4
c1355	0.74	372.0	0.65	648.0	0.75	93.6
struct	0.76	1780.0	0.67	3180.0	0.77	222.0
c3540	0.75	1800.0	0.60	5760.0	0.75	502.0

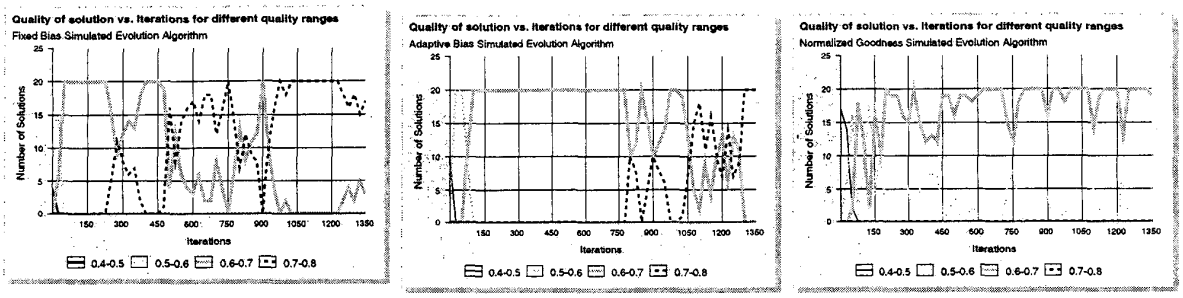
Table 1: Comparison of solution quality ($Q \in (1,0)$) and algorithm execution times (T in minutes) of best fixed bias, normalized goodness (N.G.) and adaptive bias (A. Bias) SE algorithms. A higher value of Q means better solution.

normalized goodness based SE generates worst quality solutions with excessive execution times. The reason is that this algorithm variation fails to control the size of selection set resulting in large and expensive perturbations.

The quality of solution of the adaptive bias SE is comparable to the best fixed bias SE. For all test cases, the execution time of the proposed adaptive bias SE is always a fraction of the other two approaches, fixed bias and normalized goodness SE[2].

Figure 2(a) compares the search pattern for different bias schemes. From this figure, it is clear that the quality of search for fixed bias and adaptive bias are almost identical. For the normalized goodness scheme, we observed, as expected, more frequent poor quality perturbations being performed. Further, the magnitude of these perturbations are larger than those observed for fixed and adaptive bias schemes. (see Figure 2(a)). Figure 2(b) shows that both fixed bias and adaptive bias SE require a fraction of time compared to normalized goodness SE. The behavior illustrated in these two figures is attributed to the number and type of cells selected at each iteration for perturbation. Figure 2(c) compares the cardinality of selection set. From this figure, it is clear that for Adaptive bias the selection set remains in a narrow band and slightly more than the selection set size of the best fixed bias scheme. The reason for narrow band is that when average goodness is low, high bias maintains a limited size of selection set. On the other hand, normalized goodness SE algorithm always has a larger selection set than any of the other schemes. Further, the selection set is always excessively large making the state space walk unnecessarily random rather than evolutionary. This results in unnecessary perturbations of the well placed cells, reducing the quality of solution and increasing the execution time. Figure 3 shows the average goodness

[2]The run time of fixed bias SE is the sum of execution times of all trial runs required to choose the value of bias. On average, five trial runs were required.



(a) (b) (c)

Figure 4: Comparison of different bias schemes. Quality of solution against iteration of the algorithm for different quality ranges: (a) fixed bias; (b) adaptive bias; and (c) normalized goodness SE algorithm.

of selected elements for these three schemes. Adaptive bias scheme has lowest average goodness of selected cells. It means that it selects more unfit elements than other two schemes. The normalized goodness algorithm selects high average goodness cells.

Figure 4 shows the quality of solutions found by the three bias schemes. These graphs, plot the number of solutions found (in a step of 20 iterations) by these schemes for different solution quality ranges. For clarity, only four solution quality segments are shown. According to Figures 4(a), during the initial stages of the search, fixed bias scheme finds more solutions in low quality range (0.4-0.5, 0.5-0.6). As the algorithm progresses, it finds more good quality solutions. The same behavior is observed with the adaptive bias scheme in Figure 4(b). Hence, both algorithms exhibit tendency of narrowing the search to fitter solution subspaces. The perturbations are not of excessive magnitude as to cause deterioration in quality of solutions. In contrast, the normalized goodness algorithm was unable to find any solutions in the high quality range (0.7-0.8). This shows the limitation of this algorithm variation to come out of local minima.

5. CONCLUSION

This work proposes a new adaptive bias scheme where bias value automatically evolves as the search progresses. For each iteration of the SE algorithm, bias is set equal to $1 - G$, where G is the average population goodness of current iteration. This makes the algorithm more adaptable to the overall quality of solution. It also reduces the size of the selection set in early iterations leading to a considerable saving in the execution time. Further, *Bias* is no longer an algorithm parameter that must be tuned for each problem instance. Experimental results with VLSI placement benchmark test cases indicate that the proposed scheme results in a robust and truly general algorithm for combinatorial optimization.

Acknowledgments

Authors acknowledge King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, for all support.

6. REFERENCES

[1] Ralph M. Kling and Prithviraj Banerjee. Empirical and Theoretical Studies of the Simulated EvolutionMethod Applied to Standard Cell Placement. *IEEE Transactions on Computer-Aided Design*, 10(10):1303–1315, October 1991.

[2] Yuh-Sheng Lee and A.C.-H. Wu. A performance and routability-driven router for FPGAs considering path delays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16:179–185, February 1997.

[3] Yau-Hwang Kuo, Shaw-Pyng Lo, P. Dewilde, and J. Vandewalle. Partitioning and scheduling of asynchronous pipelines. In *CompEuro '92. Computer Systems and Software Engineering*, pages 574–579, May 1992.

[4] Y. L. Lin, Y. C. Hsu, and F. H. S. Tsai. SILK: A Simulated Evolution Router. *IEEE Transactions on Computer-Aided Design*, 8(10):1108–1114, October 1989.

[5] Sadiq M. Sait and Habib Youssef. *VLSI Physical Design Automation: Theory and Practice.* McGraw-Hill Book Company, Europe (also co-published by IEEE Press), 1995.

[6] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms and their Application to Engineering.* IEEE Computer Society Press, 1999.