Adaptive Bias Simulated Evolution Algorithm

Habib Youssef   Sadiq M. Sait   Hussain Ali

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

Abstract: Simulated Evolution (SE) algorithm is an iterative stochastic algorithm designed by Kling and Banerjee [1] for combinatorial optimization problems. SE iteratively works on a single solution of movable elements, where each element is characterized by a goodness measure which estimates its fitness with respect to current state. A new solution is evolved from current solution by relocating some of the movable 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 algorithm resorts to a *Selection Bias* parameter. This parameter has major impact on the algorithm run-time requirement and the quality of the solution subspace searched [2]. Kling and Banerjee observed that low values of Bias (in the range of -0.2 to 0.2) give best results, with the exact value to be decided from trial runs. The trial runs would be required for each problem instance.

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 preset value. The proposed adaptive bias SE algorithm produces results of comparable quality to those obtained with the best fixed selection bias, while avoiding the expensive trial-runs time consuming trial runs. Also, the adaptive bias and the fixed bias schemes are compared to goodness normalization, another approach proposed in reported literature.

1 Introduction

Kling and Banerjee proposed Simulated Evolution (SE) as a general algorithm [1] for the solution of combinatorial optimization problems. SE considers the solution as a population of movable elements. During the course of the algorithm it tries to move these elements to their optimum positions such that the overall cost of the solution is reduced.

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 to current cost estimate. 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. These selected elements are identified as selection set and removed from the solution. These selected elements are one by one reassigned 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 is not possible as it requires the knowledge of optimum cost. Therefore a rough estimate of goodness is used. 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. Futher, a high positive value of bias decreases the probability of selection of elements thus reducing the size of selection set.

The bias value has an effect on the execution time and the quality of final solution [2]. A low bias value takes more execution time due to large selection sets. The quality of solution is also bad due to uncertainty created by large perturbations [2]. Similarly, for high bias values the size of the selection set is small. The quality of solution is bad due to limitations of algorithm to get out of local minima [2]. A carefully tuned bias value results in improved algorithm performance.

One way of finding a suitable bias value is to do several trial runs of the algorithm 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 proposed a **normalized goodness** scheme with zero bias value [3, 4, 5]. 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.

As a modification to the concept of fixed bias value in the SE algorithm, this work proposes **adaptive bias** which is a function of *quality of solution*. When the overall solution quality is bad, a high value of bias is used, otherwise a low value is used. The bias value changes from iteration to iteration and depends on the quality of solution. The adaptive bias automatically adjusts according to the problem state and is no longer an arbitrary or carefully tuned user defined parameter. Experiments show that the proposed bias scheme results in comparable quality of best solution while saving in execution time compared to fixed bias. It outperforms the normalized goodness scheme with respect to both quality and execution time. In this work, SE algorithm and its variations are used to find solutions to the cell placement problem of VLSI design. Placement is an NP-Hard combinatorial optimization problem [6] which can be formulated as follows [7]: *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 interconnect wire-length has been widely used as the objective of VLSI placement.

The rest of this paper is organized as follows. Section 2 covers different bias schemes for SE algorithm. The proposed *Adaptive bias simulated evolution* approach is described in Section 2.2. Experimental results are given in Section 3. We conclude in Section 4.

2 Selection Bias Schemes in SE algorithm

2.1 Fixed Selection Bias

The original SE algorithm proposed by Kling and Banerjee uses fixed selection bias. Selection *bias* (B) is used to compensate errors made in estimation of goodness measure as well as to control the magnitude of perturbation. Figure 2 shows the effect of *bias* on various aspects of SE algorithm, namely (a) quality of solution in terms of wire-length cost of best solution, (b) runtime, and (c) number of selected elements for relocation at each iteration. A carefully tuned *bias* value results in good solution quality and reduced execution time [2]. However it requires many trial runs of the algorithm to tune bias value.

The effect of bias on the quality of final solution can be described as follows. When a low bias value is used, a large number of elements are selected in each iteration. This will decrease the algorithm ability to place elements optimally due to uncertainty about too many unplaced elements. It will also increase the execution time of the algorithm. In contrast, when bias is very high, the algorithm ability to get out of local minima is restricted because badly placed elements have reduced chances of selection. Thus, the size of the selection set, execution time as well as the quality of solution space searched is reduced [2].

2.1.1 Normalized Goodness

One solution to the problem of finding suitable bias value in advance is to normalize individual goodness [3, 4, 5] and use zero bias value. The objective of the normalization is to spread the goodness within *Upper* and *Lower* bounds where $Upper + Lower \leq 1.0$. The advantage of this approach is that the SE algorithm becomes independent of bias value. Yuh et. al. [3] have normalized individual *score* in the range 0.05 and 0.95. Using their normalization method for our problem, we modified individual cell goodness as follows.

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

Where

$g_i =$	actual goodness of cell i
$g_i^{\min} =$	$\min_i(g_i)$
$g_i^{\max} =$	$\max_i(g_i)$
$g_i' =$	normalized goodness of cell i.

Using the above equation, the best cell in a solution will have a goodness of 0.95. On the other hand, the worst cell will be identified with the goodness of 0.05. As we will see in Section 3, normalized goodness fails to control the size of selection set resulting in huge perturbations.

2.2 Proposed Adaptive Selection Bias

As a modification to the concept of fixed bias value in the SE algorithm, this work proposes an **adaptive bias** scheme which will be automatically estimated by the algorithm as a function of *quality of solution*. When the overall solution quality is bad, a high value of bias is used, otherwise a low value is used. The bias value will change from iteration to iteration and will depend on the quality of solution. Average cell goodness of a solution is a measure of how all the cells are near their optimum positions. The bias value is a function of average goodness i.e., for k^{th} iteration of the algorithm, bias B_k will be computed as follows.

$$B_k = 1 - G_{k-1} \tag{2}$$

where G_{k-1} is the average goodness of all the cells at the end of $(k-1)^{st}$ iteration of the SE algorithm. 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. For bad 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 larger perturbations.

3. For good quality solutions, the average goodness is high. Therefore, a low bias value is used. This will result in the selection of a sufficient number of cells which will protect the algorithm from early convergence.

3 Results

In this section we experimentally compare the effect of different bias schemes on SE algorithm. This comparison covers three selection bias strategies namely **fixed**

bias, **normalized goodness** with zero bias and **adaptive bias**. SE algorithm is applied to solve the VLSI cell placement problem with the objective of minimizing wire-length cost. The tests are carried out on ISCAS-89 benchmark circuits. Initial solutions are randomly generated and algorithms are executed for a fixed number of iterations. This comparison is based on the quality of the solution and algorithm execution time. The quality of solution $Q = 1 - NC$ where $NC \in (0,1)$ is normalized cost of the solution with respect to wire length. In order to gain insight into these schemes, the cardinality and average goodness of selection sets are also compared. Because the bias is used to control the size of selection set which in return affects the quality of solution and execution time.

Table 1 compares the quality of final solution and execution time of SE algorithm with different bias schemes. The normalized goodness based SE algorithm generates worst quality solutions with excessive execution times. It is due to the fact that this algorithm fails to control the size of selection set resulting in lower efficiency of the algorithm.

The quality of solution of the adaptive bias SE algorithm is comparable to the best fixed bias SE algorithm. For some test cases, the execution time of the adaptive bias SE algorithm is slightly more than the best fixed bias SE. However, if we consider the time spent in trial runs of the SE algorithm to find the best bias fixed value, then adaptive bias outperforms the fixed bias SE algorithm. At least 4 trial runs with different bias values were required to identify the suitable bias for each circuit. Time spent for these trial runs are not included in the execution time of fixed bias algorithm. The other advantage of adaptive bias is that bias is no longer a user specified value.

Figure 3(a) compares the search pattern for different bias schemes in the SE algorithm. For fixed bias scheme, the best bias value is determined after several runs of SE algorithm with different bias values. From this figure, it is clear that the quality of search for fixed bias and Adaptive bias are almost identical. The normalized goodness scheme results in low quality solutions throughout the execution of the algorithm, while requiring excessive execution times (see Figure 3(b)). The behavior of these schemes is attributed due to the number and type of cells selected at each iteration for perturbation. Figure 3(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. Similarly when average goodness improves, a low bias does not allow the size of selection set to considerably decrease. On the other hand, normalized goodness SE algorithm always has a bigger selection set than any of the other schemes. It is due to the fact that even the best placed cell has a non-zero probability of selection. This results in unnecessary perturbations of the well placed cells, reducing the quality of solution and increasing the execution time. Figure 3(d) shows the average goodness of selected elements for these three schemes. Adaptive bias scheme has low 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 these schemes. In these graphs, quality of solution is measured by plotting the number of solutions found (in a step of 20 iterations) by these schemes for different quality ranges. For clarity, only four solution quality segments are shown. According to Figures 4(a), during the initial stages of 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 limitation of algorithm to come out of local minima.

4 Conclusion

SE algorithm, as initially proposed by Kling and Banerjee, is heavily influenced by a *Selection Bias*, a difficult to tune algorithm parameter. This paper investigated the effect of different selection bias schemes on the quality of solution produced by the simulated evolution (SE) algorithm. We examined two schemes reported in the literature. The first one uses a fixed bias value [2] and the second one avoids using the bias parameter by normalizing the goodness measure to a desired interval [3, 4, 5]. The best fixed bias value results in reduced solution cost as well as reduced execution time. However, finding out the best bias value requires several trials with different bias values. On the other hand, the normalized goodness avoids the use of bias value altogether. This scheme performs well for small problems. As the size of the problem increases, it causes excessive execution time and deteriorates quality of solution.

We proposed adaptive bias scheme where bias value is a function of average cell goodness. For each iteration of the SE algorithm, bias is set equal to $1 - G$, where G is the average population goodness of previous 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. It also saves the algorithm from getting trapped in local minima by progressively reducing the bias value as the search zooms on fitter solution subsapces. 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.

References

[1] Ralph M. Kling and Prithviraj Banerjee. ESP: Placement by Simulated Evolution. *IEEE Transactions on Computer-Aided Design*, 8(3):245–255, March 1989.

[2] Ralph M. Kling and Prithviraj Banerjee. Empirical and Theoretical Studies of the Simulated

Circuit	Fixed Bias		Normalized Goodness		Adaptive Bias	
	Q	T	Q	T	Q	T
highway	0.53	0.1	0.48	1.0	0.48	0.1
fract	0.65	3.0	0.64	5.4	0.64	1.0
c499	0.64	3.5	0.64	20.4	0.63	2.8
c532	0.67	10.4	0.69	24.4	0.65	5.6
c880	0.77	12.4	0.75	72.0	0.77	17.4
c1355	0.74	74.4	0.65	648	0.75	93.6
struct	0.76	356.0	0.67	3180	0.77	222.0
c3540	0.75	360.0	0.60	5760	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 and Adaptive bias SE algorithms. A higher value of Q means better solution.

EvolutionMethod Applied to Standard Cell Placement. *IEEE Transactions on Computer-Aided Design*, 10(10):1303–1315, October 1991.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

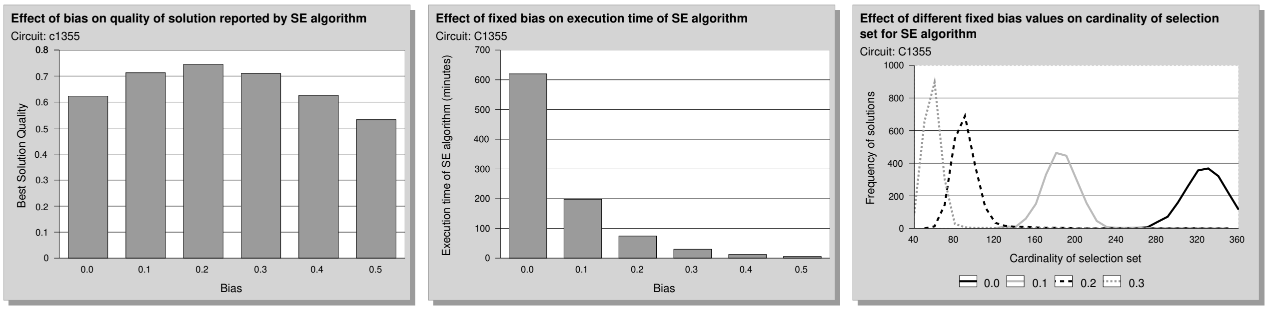
[7] Y. Saab and V. Rao. Stochastic Evolution: A Fast Effective Heuristic for some Generic Layout Problems. In *27th ACM/IEEE Design Automation Conference*, pages 26–31, 1990.

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

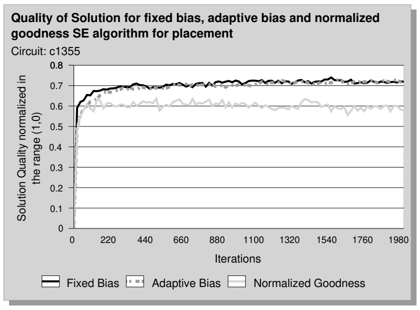
Until Stopping Condition is satisfied
Return Best solution.
End $(Simulated\_Evolution)$

Figure 1: Structure of the simulated evolution algorithm.

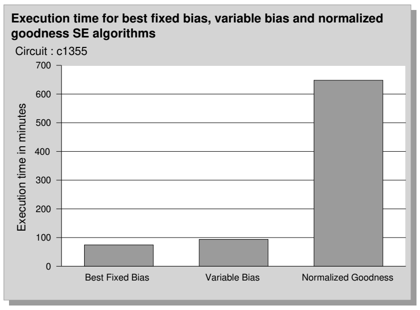


(a) (b) (c)

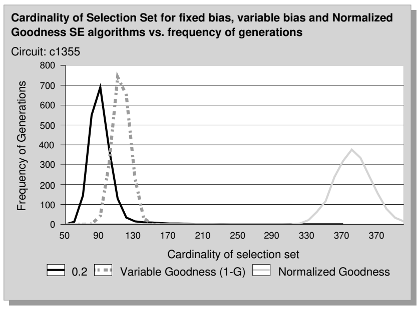
Figure 2: Effect of bias on the Simulated Evolution algorithm. (a) quality of solution generated (b) execution time of the algorithm (c) cardinality of selection set against frequency of solutions



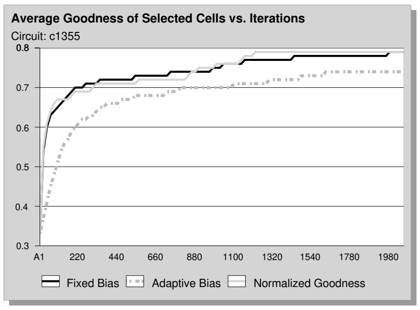
(a)



(b)

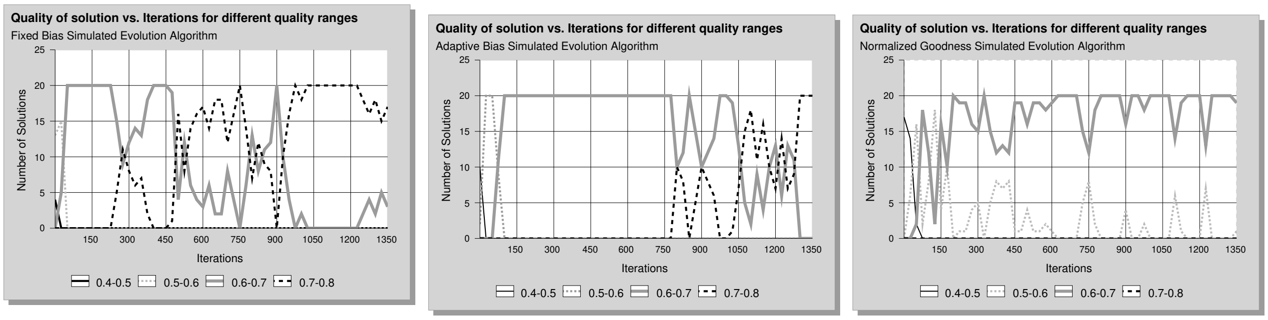


(c)



(d)

Figure 3: Comparison of different bias schemes (a) current wire-length cost (b) execution time (c) cardinality of selection set against frequency of solutions for different bias schemes for the SE algorithm.



(a) (b) (c)

Figure 4: Comparison of different bias schemes (a) current wire-length cost (b) execution time (c) cardinality of selection set against frequency of solutions for different bias schemes for the SE algorithm.