

Relaxation and Clustering in a Local Search Framework: Application to Linear Placement *

Sung-Woo Hur and John Lillis

Dept. of Electrical Eng. and Comp. Sci., University of Illinois at Chicago

Abstract

This paper presents two primary results relevant to physical design problems in CAD/VLSI through a case study of the linear placement problem. First a local search mechanism which incorporates a neighborhood operator based on constraint relaxation is proposed. The strategy exhibits many of the desirable features of analytical placement while retaining the flexibility and non-determinism of local search. The second and orthogonal contribution is in netlist clustering. We characterize local optima in the linear placement problem through a simple visualization tool – the *displacement graph*. This characterization reveals the relationship between clusters and local optima and motivates a dynamic clustering scheme designed specifically for escaping such local optima. Promising experimental results are reported.

1 Introduction

Three of the most successful and prominent paradigms for solving physical design problems in CAD/VLSI are **constraint relaxation**, **local search**, and **netlist clustering**. This paper provides new perspectives on each of these topics via a case study in linear placement.

1.1 Background

Constraint relaxation has provided the foundation of numerous university and commercial approaches to the cell placement problem. These techniques are often referred to as “analytical placement” and generally adopt the following or similar philosophy. First, the “slot constraints” of the problem are released resulting in a continuous space optimization problem which can be solved optimally and efficiently.¹ A solution to this relaxed formulation results in a physically infeasible placement with cell overlap; this placement is then refined in a top-down manner (typically by recursive partitioning of the layout area) to eventually converge on a physically feasible solution. The intuitive appeal of such approaches is that the relaxation solves a true global optimization problem taking into account the entire netlist simultaneously. The methods also tend to be quite

*This work was supported by the UIC Campus Research Board and the Design Automation Conference Scholarship Program

¹There are several possibilities with respect to the particulars of the continuous formulation – e.g., quadratic placement (e.g., [1]), “linearized” quadratic placement [2], linear programming based relaxations [3] and quadratic placement using spectral methods (e.g., [4]).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 99, New Orleans, Louisiana
©1999 ACM 1-58113-092-9/99/0006..\$5.00

computationally efficient. (As an aside, min-cut based placers (e.g., [5]) can also be considered members of this general class.²)

The second paradigm of **local search** optimizes via repeated perturbation of a current solution (typically, but not always, physically feasible). A final optimized solution is converged upon via a sequence of such perturbations. Typically, perturbations (or moves) are very simple – e.g., pairwise exchange. The candidate moves define a neighborhood structure for the problem – each configuration (placement) having a set of neighboring configurations which can be reached in a single move. The intuitive appeal of this kind of approach is multi-faceted. First, the entire solution space is reachable via some sequence of moves (assuming a reasonable neighborhood operator). Second, the technique can be made non-deterministic via randomization allowing natural exploitation of additional CPU resources (relaxation-based techniques are, by and large, much more deterministic in nature). Further, in the particular case of Simulated Annealing (SA) (e.g., [7]) – perhaps the most celebrated local search paradigm for cell placement – there are some theoretical results (e.g., [8]) indicating that, with a proper cooling schedule, SA converges to a global optima with probability approaching 1.³ Yet, perhaps the most convincing testimony to the local search paradigm is its remarkable success in practice.

The third paradigm – **netlist clustering** – can be considered orthogonal to the first two. The motivation for netlist clustering varies significantly from paper to paper. A typical reason for clustering is simply to more effectively deal with huge designs – by pre-processing the netlist and creating a clustered netlist, the problem size becomes more manageable. There are a number of other engineering-oriented reasons for clustering – e.g., in a design with widely varying cell sizes, a clustering step is frequently used to create clusters of roughly equivalent size (thereby enabling the use of cell-oriented algorithms on the clustered netlist). Some recent results in partitioning [9, 10] have also successfully applied clustering technology to obtain state of the art results in impressive CPU time. However overall, it appears that clustering technology is not quite as mature as some other techniques with little consensus on appropriate clustering metrics and on the underlying objectives of clustering itself.

1.2 Contributions

The results in this paper touch on all three of these topics: constraint relaxation, local search and clustering. The contributions are summarized as follows.

- First, we propose a local search mechanism in which

²However, the relaxation of slot constraints for min-cut is not continuous – rather many slots are replaced with two (or four in the case of quadrisecting [6]) – and the relaxed problem is not solved optimally because of the NP-completeness of graph partitioning.

³Unfortunately, the rate of convergence is not, in general, polynomially bounded.

the neighborhood operator itself is much more sophisticated and directed than those typically used. The operator is itself based on a linear programming relaxation of the problem where a subset of the cells are “mobile” and the remaining cells are fixed by the current placement. An efficient network flow algorithm is used to solve the linear programming (LP) formulation. Generally speaking, a single move in this scheme is as follows: extract (by some randomized means) a sub-circuit from the netlist; solve the LP relaxation where all cells outside the sub-circuit are at fixed positions as determined by the current placement; heuristically legalize the resulting relaxed placement; evaluate the new placement and accept or reject. The technique adopts a more analytical and global view of the problem while maintaining the desirable features of the local search paradigm. Experimental evidence is presented indicating the potential of such a unification of analytical methods and search-based methods.

- Second, we present studies characterizing local optima in the linear placement problem and their relation to circuit clusters. We have devised a simple visualization tool – the *displacement graph* – for characterizing the differences between two placements. By studying the differences between known excellent placements and mediocre local optima it is revealed that such local optima typically have successfully found many appropriate clusters of cells, but that these clusters are not globally placed correctly. A local search algorithm working on the flat netlist is unlikely to uncover this structure and as a result it is usually very difficult to improve such solutions (i.e., they are local optima). However, via a simple dynamic clustering strategy based on the current linear placement, such global structure can often be revealed, allowing us to escape the local optima. The key points of the clustering strategy are as follows.

- Clusters should be derived from the current placement. A similar philosophy has been adopted by Saab [11]. The idea is that good placement algorithms do an excellent job of finding good clusters; instead of finding clusters via a pre-process, we let the placer do the work.
- The strategy should be dynamic. While simple strategies based on wiring density seem to identify good clusters in a given placement, it seems unwise to rely on just a few such decisions. As a result, the algorithm repeatedly clusters and flattens the circuit optimizing in each phase via the local search algorithm.

We have dubbed the resulting strategies **RBLS** (“rebels”) for *Relaxation Based Local Search* and **RBLS/C** for the dynamic clustering version. These techniques have produced very promising experimental results: new best known results have been found for every circuit in a set of benchmarks used in a recent series of papers on linear placement [12, 13]. Additionally, the computational overhead of the techniques is shown to be reasonable.

The remainder of the paper is organized as follows. In the next section, we define some necessary terminologies. In section 3, we present the relaxation based local search algorithm. Section 4 presents our dynamic clustering strategy. Section 5 presents the overall *RBLS/C* algorithm. Section 6 presents experimental results on the linear placement with MCNC benchmarks, followed by conclusions in section 7.

2 Preliminaries

We model a netlist by a hypergraph $G(V, E)$, where V is a set of cells and E is a set of nets. A hyperedge $e \in E$ is a subset of 2 or more cells in V (i.e., $e \subseteq V$). Each cell corresponds to a component of a circuit and each net represents a common signal among its constituent cells.

For a linear placement P , let $P[i]$ denote the cell placed in location i and $P^{-1}[v]$ the location of cell v with the assumption that each cell has unit-size. A placement P is *legal* if $P : N \Rightarrow N$ is one-to-one function. Let e_i be a set of cells which are connected to a net i . Then, the length of net i , $len(e_i)$, is defined as

$$len(e_i) = \max_{u, v \in e_i} |P^{-1}[u] - P^{-1}[v]|$$

and the linear placement problem as

$$\min \sum_{i=1}^{|E|} len(e_i) \text{ over all legal placement } P.$$

3 Relaxation-Based Local Search

3.1 Overview

The top-level local search strategy used in RBLS is quite traditional. From the current solution we sample a neighboring solution and move to that solution if the objective (wire length) is improved. If no improvement is seen for k (a given parameter) consecutive moves, the search terminates. The novelty of RBLS is in how we generate neighboring solutions. This process is summarized as follows:

- **Sub-circuit Extraction:** Given a parameter m , extract a sub-circuit $M (\subseteq V)$ where $|M| = m$. M will be called the set of “mobile nodes”. From M we determine the *fixed* node set F : the set of nodes in $(V - M)$ which are directly connected to a member of M via some net. The extracted sub-circuit consists of node set $F \cup M$ and net set E' induced by M , i.e., $G' = (F \cup M, E')$.
- **Optimal Relaxed Placement:** Optimally place each member of M ignoring slot constraints under a linear programming relaxation of the problem. Note that the relative order of cells in F influences this solution.
- **Placement Legalization:** Resolve cell overlap to obtain a physically feasible placement.

Each of these steps is detailed in the following subsections.

3.2 Sub-circuit Extraction

To extract a sub-circuit $G' = (V', E')$, where $V' = F \cup M$, we extract M first. The simplest method for extracting a subset M would be a random selection. However, such a simple scheme would result in many disconnected components and the resulting optimization problem does not capture much of the interaction between cells. Therefore, we have adopted the following randomized scheme which produces a set of connected components. Experiments have shown this scheme to be superior to simple random selection. First, given a parameter m , mobile nodes are selected as follows:

1. $M \leftarrow \emptyset$ // mobile node set
2. $C \leftarrow \emptyset$ // Set C has candidate nodes for mobile nodes
3. Extract a node $v \in V \setminus M$ at random and $M \leftarrow M \cup \{v\}$
4. $A \leftarrow \{u \in e \mid v \in e, \forall e \in E\}$ // nodes adjacent to v
5. $C \leftarrow C \cup A \setminus M$
6. If $|C| = 0$ goto step 3
7. Extract a node $v \in C$ at random and $M \leftarrow M \cup \{v\}$
8. Repeat step 4 - 7 while $|M| < m$

Once M is extracted, we determine the fixed node set F and the “active nets” E' (all nets which influence the relaxed placement problem for M) as follows: $E' = \{e_i \mid M \cap e_i \neq \emptyset\}$. $F = \{v \mid v \in e_i \setminus M, \text{ where } e_i \in E', \text{ and } v \text{ is either left- or right-extreme node of } e_i \text{ in the current placement } P\}$.

Figure 1 shows an example of extracted sub-circuit and current placement P .

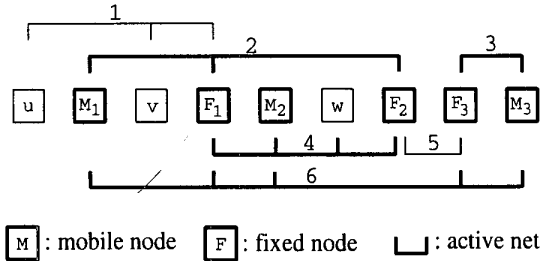


Figure 1: An example of the current placement P and a sub-circuit extracted from P . Total wire length of P is 20.

3.3 LP-Formulation of a Relaxed Placement

Given a set of mobile nodes M , we can derive a simple linear program for optimally placing M . Such a linear program will produce an x -coordinate x_v for each mobile node $v \in M$. The LP is of course influenced by the locations of the fixed nodes F ; let X_v be the location of a node $v \in F$ in a given feasible placement P , i.e., $X_v = P^{-1}[v]$. Then the LP relaxation can be stated as follows.

$$\begin{aligned} \min \sum_{e \in E'} (r_e - l_e) \text{ s.t.} \\ l_e \leq x_v \leq r_e, \forall v \in e, \\ x_v = X_v, \forall v \in F \end{aligned}$$

The dummy variables r_e and l_e in the formulation give the leftmost and rightmost ends of net e .

To illustrate the relaxation process, suppose we have a given legal placement P and sub-circuit as shown in Fig 1. Figure 2 shows a possible solution of LP (optimal solutions not being necessarily unique).

In early experiments, a public domain LP-solver [14] was used but it proved to be an unacceptable bottleneck. Fortunately, it was discovered that the solution of the linear program could be obtained very efficiently by using network flow techniques presented next.

3.4 Network Flow Based Algorithm

We have devised a simple network flow based algorithm for solving the LP relaxation used in our neighborhood operator. The algorithm iteratively finds minimum cuts from left to right which assign mobile nodes to bins formed by each of the fixed nodes. The resulting placement is an optimal solution to the LP formulation. The algorithm has also proved to be very efficient in practice.

Given an extracted sub-graph $G' = (M \cup F, E')$ and the current placement P , we use f_i ($1 \leq i \leq |F|$) to denote the

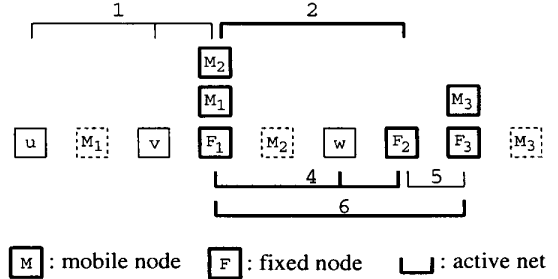


Figure 2: An example of an optimal relaxed placement, which is induced by the sub-circuit shown in Figure 1. Net 3 is not shown in the placement since its wire length becomes 0. The total wire length is 14.

i th fixed node from the left, and assume all the nodes in F are arranged according to their x -coordinates i.e., $P^{-1}[f_i] < P^{-1}[f_j]$ if $i < j$.

As illustrated in Figure 3⁴, we add additional nodes, source s and sink t and additional edges of capacity ∞ which connect fixed nodes and source/sink. For some k ($1 \leq k \leq |F|$), every fixed node f_i ($1 \leq i \leq k$) is connected to s via edge $\langle s, f_i \rangle$ and every fixed node f_j ($k < j \leq |F|$) to t via edge $\langle f_j, t \rangle$ as shown in Figure 3.

A **min-cut** is a bipartitioning (A_k, \bar{A}_k) of G' with $s \in A_k, t \in \bar{A}_k$ such that the cutsize $c(A_k, \bar{A}_k) = \sum_{u \in A_k, v \in \bar{A}_k} c\langle u, v \rangle$ is minimized, where $c\langle u, v \rangle$ is the capacity of edge $\langle u, v \rangle$.

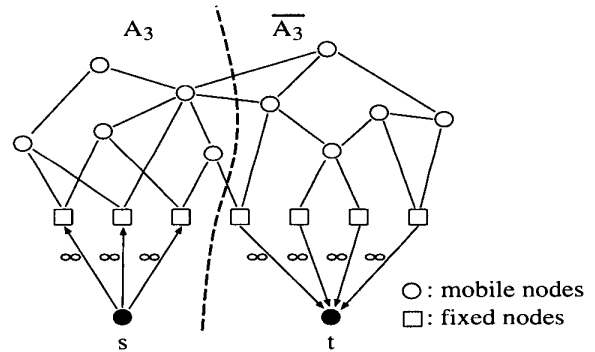


Figure 3: An example of a min-cut

Figure 4 shows the algorithm *RelaxedPlacement* (RP). In the algorithm, the set $Left$ is used to keep the mobile nodes of the current min-cut.⁵ We change directed edges between fixed nodes and source/sink one by one from the left to right. At step i , a new set of mobile nodes to be placed with fixed node f_i is identified and added to the set $Left$.

The efficiency of the algorithm is improved by maintaining residual capacities from one step to the next. As a result we are able to avoid re-computation of many augmenting paths.

Figure 5 illustrates an example of the set New after executing one iteration from the state shown in Figure 3. Mo-

⁴In this example every net is a 2-pin net and is represented by 2 opposite directed edges as a single undirected edge.

⁵We currently use a straightforward augmenting paths algorithm for finding min-cuts.

<p>Algorithm <i>RelaxedPlacement</i> (<i>RP</i>)</p> <p>Input: placement P and sub-circuit $G' = (V', E')$</p> <p>Output: relaxed placement P'</p> <p>// Initially connect all fixed nodes to <i>sink</i>. Create an edge $\langle f_i, t \rangle$ with capacity ∞, $\forall f_i \in F$; $Left \leftarrow \emptyset$; for $i = 1$ to F do { Delete edge $\langle f_i, t \rangle$; Add edge $\langle s, f_i \rangle$ with capacity ∞; $A \leftarrow$ Find leftmost min-cut using any flow algorithm; $A \leftarrow A \setminus F$; $New \leftarrow A \setminus Left$; $x_v \leftarrow P^{-1}[f_i]$, $\forall v \in New$; $Left \leftarrow A$; } <p>return P';</p> </p>

Figure 4: The algorithm *RelaxedPlacement* (*RP*). All the current residual capacities are kept for next iteration in the loop.

mobile nodes u and v become new members of *New* after modifying connectivity of the fixed node w .

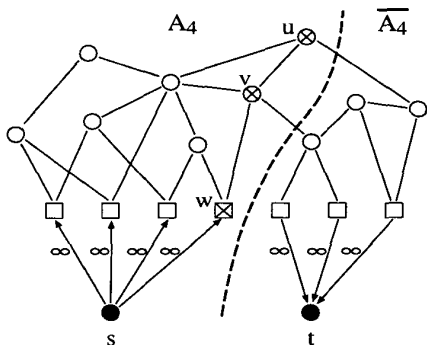


Figure 5: A new min-cut after executing one iteration of the algorithm *RP* from the state shown in Figure 3. Mobile nodes u and v become members of “New”.

Theorem 1 *The algorithm RP finds an optimal relaxed placement.*

Proof: Refer to [15]

As an aside, we note that such a binned placement is not necessarily the best in the sense that it may be more difficult to effectively legalize than other optimal solutions. Thus, techniques for finding optimal relaxed placements with more even distribution are worth studying (e.g., via finding equivalent cuts and exploiting node mobility).

We use standard constructions [16, 17] to model the hypergraph as a directed graph \tilde{G} . The hypergraph to digraph construction is summarized as follows.

- Every 2-pin net $e_i = \{u, v\}$ in G is associated with two directed edges $\langle u, v \rangle$ and $\langle v, u \rangle$ in \tilde{G} with each having capacity 1.
- Every multi-pin net e_i in G introduces two dummy nodes a, b with an edge $\langle a, b \rangle$ with capacity 1 and di-

rected edges $\langle u, a \rangle$ and $\langle b, u \rangle$ for every $u \in e_i$ with capacity ∞ .

In Figure 6 we show an example of an optimal relaxed placement which is found by the algorithm *RP* and induced by the sub-circuit shown in Figure 1. Note that the two optimal relaxed placements – one is shown in Figure 2 and the other in Figure 6 – have the same cost.

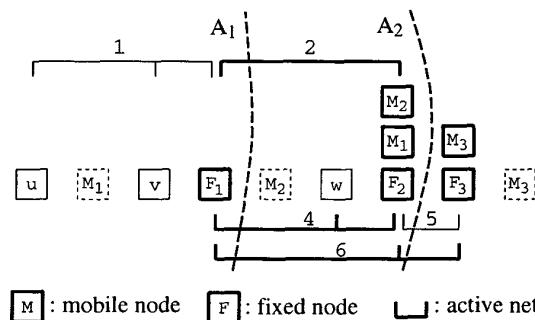


Figure 6: An example of an optimal relaxed placement found by the algorithm *RP*. This is induced by the sub-circuit shown in Figure 1. Net 3 is not shown in the placement since its wire length becomes 0. The total wire length is 14.

Empirical results have shown remarkable speedups versus the LP-solver. For example, an instance with 5000 mobile nodes and roughly 1000 fixed nodes can be solved in less than 8 seconds on a 167 MHz Sun Ultra-Sparc 1 while the LP-solver takes about 55 minutes.

3.5 Placement Legalization

A central problem in relaxation-based methods is the resolution of cell overlaps or *legalization*. A relaxed placement may result in empty spaces and overlapping nodes that should be resolved to get a physically feasible placement.

Our legalization scheme is quite simple. We use a *force value* to get a relative order among coincident nodes. Let $S(\subseteq M)$ be a set of coincident nodes and n_i a set of nets adjacent to node i . Suppose the position for each node in S in the relaxed placement is x_s . Let l_e be the left extreme end of a net e and L_i be a subset of n_i such that $L_i = \{e \mid e \in n_i \text{ and } l_e < x_s\}$. Similarly, let r_e be the right extreme end of a net e and R_i be a subset of n_i such that $R_i = \{e \mid e \in n_i \text{ and } r_e > x_s\}$. Then, the force value d_i for node i in S is computed as $d_i = |R_i| - |L_i|$.

Using the force values, we decide the relative order of nodes in S , e.g., the greater force value a node has, the greater x-coordinate will be assigned to it. For the nodes having the same force value we randomly rearrange them. After determining the relative order of nodes in S , we place them at the position of the associated fixed node while moving other nodes keeping their relative order. Figure 7 shows the corresponding legalized placements induced by the relaxed placement which is obtained by the *RP* algorithm as shown in Figure 6.

4 Dynamic Clustering

Effective methods for escaping local optima are essential in most successful local search schemes. We propose a simple dynamic clustering technique for precisely this purpose.⁶ To

⁶We will use the notion of local optima somewhat loosely to indicate a solution which has not improved significantly in a “long time.”

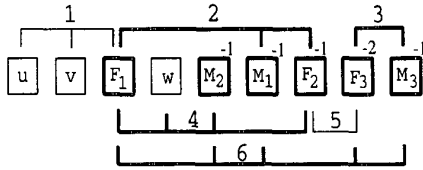


Figure 7: An example of a legalized placement induced by the relaxed placement which is shown in Figure 6. The force value for each coincident node is shown on the shoulder. The legalized placement has wire length of 18 resulting in 10% improvement over the previous legalized placement shown in Figure 1.

the best of our knowledge, this intimate relationship between clusters and local optima has not previously been studied in the literature.

Our clustering technique was motivated by an effort to characterize local optima in the linear placement problem via a simple visualization tool we call a *displacement graph*. Suppose P_g is a relatively good placement and P_b a mediocre one. Also assume that each has converged to local optima. If we consider P_g as a reference placement, *displacement* of P_b for each location i with respect to P_g is defined as

$$D[i] = P_b^{-1}[P_g[i]] - i.$$

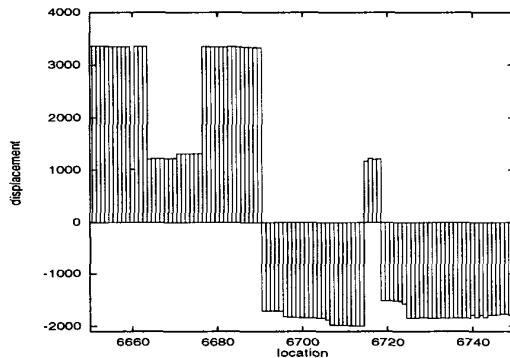


Figure 8: Displacement graph obtained from two local optima placements for circuit s38584. Mediocre solution has wire length of 1,756,332, while good reference placement has wire length of 1,253,972.

Figure 8 shows a small range of *displacement graph* so details can be seen where we plot i versus $D[i]$. A positive(negative) value indicates that the cell at that position in P_g is placed to the right(left) in P_b . It is striking how we can easily identify “plateaus” in the plot with sharp transitions between plateaus. A plateau indicates that the bad placement seems to correctly group many cells into the right clusters, but that these clusters are not correctly placed in the global scheme of things. This seems to be a signature of a local optimum – i.e., it appears that almost any neighborhood structure would have a hard time uncovering this global structure. This has led us to a dynamic clustering technique for escaping local optima.

Suppose a placement has converged to a local optima. We know that the placement might have good clusters which correspond to plateaus in a displacement graph (versus some hypothetical “good” solution). The goal will be to identify

these plateaus dynamically. If this can be done, we will have an effective tool for escaping local optima. Said another way, improvement in solution quality in the clustered circuit’s search space becomes more likely as is shown experimentally later in this section. Of course, since a good reference placement is not available to help us identify plateaus as clusters, we must resort to other schemes to heuristically estimate cluster boundaries. A simple conjecture is that there is a correlation between transitions in wiring density (number of wires) in the bad placement and the edges of plateaus. Our experiments indicate some truth to this conjecture and such a heuristic has become the basis for our clustering strategy.

We derive clusters from placement P as follows: given two parameters L and U , and the current placement P , scanning P from the left to the right it clusters a block of nodes using the density values such that each block size is between L and U and the boundary of the next block is the point at which the density value is the smallest among $d[p+L] \cdots d[p+U]$, where $d[i]$ is the density value of position i and p is the boundary position of the previous cluster.

Using the clustering algorithm we generate a clustered circuit and a clustered placement P_c based on P . Each cell in the clustered circuit is considered to have unit-size. Given P_c , the relaxation based local search algorithm of Section 3 is applied until a convergence criteria is met and then it is flattened. While flattening P_c , each clustered node is examined to see whether the reversed order may result in a better solution, and if so, the block of clustered nodes is placed in reverse order in the flat placement.

Note that since the clustering strategy is dynamic and clustering is always based on the current solution, one clustered circuit is different from previous and subsequent clusterings of the netlist.

Figure 9 illustrates the effectiveness of the technique in escaping local optima. The figure shows a snapshot of a run of the algorithm (CPU time versus wire length). Up to the first transition point (a near local optimum) the placement was optimized in flat mode. At this point, we cluster the circuit and continue the run on the clustered circuit. Wire length in the graph always gives the wire length of the flat placement implied by the clustered placement. After clustering, the result shows a sharp reduction in wire length. When there seems no significant improvement on clustered circuit, we flatten it and continue to optimize the flat circuit. This cycle is repeated and shows the remarkable effectiveness of the dynamic clustering strategy.

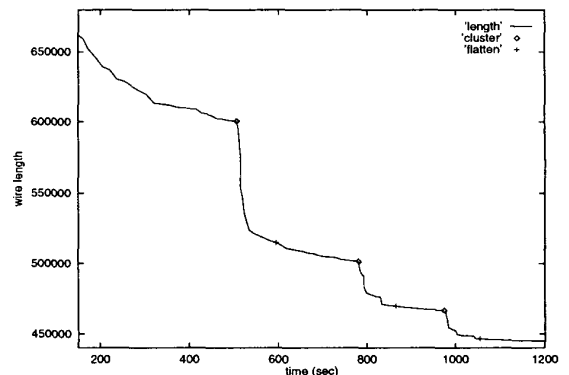


Figure 9: Effect of Dynamic Clustering. The figure shows part of a run for circuit s15850 where we toggle between flat and cluster mode. Before clustering, optimization of the flat placement has nearly converged.

5 Overall Algorithm

Fig. 10 shows the overall procedure of our relaxation-based local search algorithm with dynamic clustering, *RBLs/C*. The algorithm can be summarized as follows. First, an initial placement is generated. We then repeatedly cluster and flatten the circuit, performing local search on both flat and clustered solutions. This is done until some convergence criteria are met. Throughout the algorithm, there is also some freedom in assigning the control variables (e.g., sub-circuit size for RBLs and target cluster size).

Algorithm <i>RBLs/C</i>
Input: circuit information and control values: $i_0 \dots i_3, \alpha, \beta, \gamma, \delta, \epsilon, \mu, \omega$
Output: placement P
<pre> $m \leftarrow i_0; m_c \leftarrow i_1;$ $Imax \leftarrow i_2; Imax_c \leftarrow \alpha * Imax;$ $L \leftarrow i_3; U \leftarrow \beta * L;$ $phase \leftarrow 1;$ // used to change L and U $P \leftarrow InitialPlacement();$ while (TRUE) do { $P \leftarrow LocalSearch(P, m, Imax);$ $P_c \leftarrow Cluster(P, L, U);$ $P_c \leftarrow LocalSearch(P_c, m_c, Imax_c);$ $P_f \leftarrow Flatten(P_c);$ $Imprv \leftarrow (len(P) - len(P_f))/len(P);$ if ($Imprv < \epsilon$) return $P;$ else $P \leftarrow P_f;$ // Change control values $Imax \leftarrow Imax + 1;$ $Imax_c \leftarrow \alpha * Imax;$ $m \leftarrow \gamma * m;$ $m_c \leftarrow \delta * m_c;$ if ($phase = 1$) { // increase L $L \leftarrow \mu * L;$ if ($L \geq \omega$) $phase \leftarrow -1;$ } else{ // decrease L $L \leftarrow L/\mu;$ if ($L < 2$) $phase \leftarrow 1;$ } $U \leftarrow \beta * L;$ } </pre>

Figure 10: Overall procedure of the *RBLs/C* algorithm

We use several control variables to enable the algorithm to find high quality solution more efficiently. Values for them are changed based on either the circuit size, solution quality, the number of iterations, or the netlist type (flat or clustered).

The control variables m and m_c are used to determine the sub-circuit size of flat and clustered nets, respectively. Initial size of a sub-circuit (i_0 and i_1 are used for each type) is proportional to the circuit size and it is gradually decreased to a lower bound of 10. $Imax$ and $Imax_c$ are used for the maximum number of moves in the local search algorithm without improvement for flat and clustered circuits, respectively. Those values are monotonically increasing by setting $\alpha > 1$. As solution quality improves, experiments show that selecting smaller sub-circuits and giving more chances to explore neighbors is more effective. L and U are used to determine a target block size for clustering. The average block size in clustered circuits is dynamically changed by varying

L and U . L increases up to ω in the first phase and decreases to 1 in next phase. We alternate increasing and decreasing phases. By this way we have more chances to find globally right locations for different clusters on different placements.

6 Experimental Results

We have implemented the *RBLs/C* algorithm and tested it on a standard set of benchmarks on a 167 MHz Sun Ultra-Sparc 1.

As described, there are a number of control parameters to the *RBLs/C* algorithm. While the strategy for initializing these values is not central to our contributions, for completeness, we describe our current strategy in the following. The strategy is based on experimental experience and intuition. The strategy may evolve over time as we gain more insight into their effects; nevertheless, the current approach does seem reasonable. For the initial sub-circuit size we set i_0 to $0.8 * |V|$ and i_1 to $0.02 * |V|$. We set γ to approximately 0.25 ~ 0.3 and δ to 0.4 ~ 0.6. We use 10 as a lower bound on both m and m_c . We set i_2 – the initial value for $Imax$ – to 5. We set α to 3, i.e., for a clustered circuit the local search algorithm will allow 3 times as many consecutive failed moves before exiting.⁷ L is initially assigned 2 via i_3 . By setting $\mu = 2$ it is doubled up to ω , which is usually assigned 128, in increasing phase and cut by half in decreasing phase per each loop. To determine the upper bound on cluster size U , we set β to 1.5 ~ 2.0. When we set β to a number > 2.0 , the range of each block size of course becomes wider. Since one block is considered as a unit-size cell in clustered circuit a wider range of cluster sizes results in more error in the wire length calculation (vs. the induced flat placement). For this reason, we maintain a fairly narrow range of acceptable cluster sizes. We set ϵ to $1.0 * 10^{-5}$.

Lastly, to construct the initial placement we use the Max-Adjacency Ordering method [18]. For certain circuits (e.g., s35932) the initial placement obtained by this method is highly influential on the final solution quality while for most others the initial placement had little influence on the final solution quality. The influence of initial placement on final solution quality is a topic of on-going research.

We have run experiments on the same set of circuits tested with Sato's Simulated Quenching method (SQ) [13] and the hybrid spectral method (SLPC2) of [12]. Using the *RBLs/C* technique, new best-known result for every benchmark circuit have been found. Table 1 summarized the results. Since [13] used a different version of s1423 we do not compare our result for s1423 with that of the SQ method. The CPU time to get a reasonable solution for the larger circuits (s38584 and s38417) seems promising: typically less than 3 hours is required. For comparison, [13] reports about 8 hours on a comparable machine for s38417.

7 Conclusions

Through a case-study of the linear placement problem, this paper provides new perspectives on each of three fundamental techniques in CAD/VLSI – constraint relaxation, local search, and clustering.

First, we propose a local search mechanism in which the neighborhood operator itself is based on constraint relaxation techniques popular in analytical placement. The result

⁷This strategy seems effective since the CPU time required to visit a neighbor in the clustered circuit is generally much less than in the flat circuit. Further, since the wire-length improvement per move in the clustered circuit tends to be greater than in the flat circuit, the benefit of increasing $Imax$ in this way seems to outweigh the additional CPU time.

Ckts	Nodes	Nets	Wire Length			(% Improv. over	
			SLPC2	SQ	RBLSC/C	SLPC2	SQ
s1423	619	538	9,254	—	8,373	9.5	—
s9234	5,866	5,844	248,999	220,374	200,558	19.5	9.0
s13207	8,772	8,651	465,214	380,908	351,614	24.4	7.7
s15850	10,470	10,383	591,372	432,277	415,589	29.7	3.9
s35932	18,148	17,828	871,937	774,414	752,618	13.7	2.8
s38584	20,995	20,717	1,325,547	1,275,551	1,238,139	6.6	2.9
s38417	23,949	23,843	1,487,277	1,254,195	1,158,677	22.1	7.6
Average						17.9	5.7

Table 1: Results compared with SLPC2 and SQ methods

is a sophisticated neighborhood operator which enables very directed solution space exploration. The operator is based on a linear programming relaxation of the problem. We have devised an efficient algorithm to solve the linear programming formulation using network-flow techniques. The overall result is an optimization technique which explores the solution space with a more analytical and global view of the problem while maintaining the desirable features of the local search paradigm.

Second, we present a new dynamic clustering strategy for escaping local optima. The strategy is motivated by the studies which characterize local optima in the linear placement problem and their relation to circuit clusters. We have devised a simple visualization tool – the *displacement graph* – for characterizing the differences between two placements. By this tool, we have shown that a local optima typically have successfully found many appropriate clusters of cells, but that these clusters are not globally placed correctly. A local search algorithm working on the flat netlist is unlikely to uncover this structure and as a result it is usually very difficult to improve such solutions. However, via our dynamic clustering strategy based on the current linear placement, such global structure can often be revealed, allowing us to escape the local optima.

The experimental results prove the effectiveness of our *RBLSC/C* algorithm, which incorporates both the relaxation-based local search and the dynamic clustering techniques, by generating new best known results using less CPU-time for each circuit on which Sato [13] and Li *et al* [12] have tested.

A natural progression of this work is to adapt the techniques to the 2D case. An interesting visualization question is what is the 2D analogy of the displacement graph? Since there are 4 dimensions to consider ($x, y, \Delta x$ and Δy), creative use of color may be necessary to reveal 2D clustering information. More in-depth study of clustering techniques (including hierarchical) is also a possibility. Finally, we also are generalizing the techniques for the timing-driven placement problem.

References

- [1] C. K. Cheng and E. S. Kuh, "Module Placement Based on Resistive Network Optimization," *IEEE Transactions on CAD*, pp. 218–225, 1984.
- [2] G. Sigl, K. Doll, and F. Johannes, "Analytical Placement: A Linear or a Quadratic Objective Function?," in *28th ACM/IEEE DAC*, pp. 427–432, 1991.
- [3] M. B. Jackson and E. S. Kuh, "Performance-Driven Placement of Cell Based IC's," in *25th DAC*, pp. 370–375, 1988.
- [4] J. Frankle and R. M. Karp, "Circuit Placements and Cost Bounds by Eigenvector Decomposition," in *ICCAD*, pp. 414–417, 1986.
- [5] M. A. Breuer, "A Class of Min-cut Placement Algorithms for the Placement of Standard Cells," in *DAC*, pp. 284–290, 1977.
- [6] P. R. Suaris and G. Kedem, "Quadrisection: A New Approach to Standard Cell Layout," in *ICCAD*, pp. 474–477, 1987.
- [7] C. Sechen and A. Sangiovanni-Vincentelli, "Timber-Wolf3.2: A New Standard Cell Placement and Global Routing Package," in *23rd DAC*, pp. 432–439, 1986.
- [8] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli, "Convergence and Finite-Time Behavior of Simulated Annealing," *Advances in Applied Probability*, pp. 747–771, 1986.
- [9] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel Hypergraph Partitioning: Application in VLSI Domain," in *DAC*, pp. 526–529, 1997.
- [10] C. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel Circuit Partitioning," in *DAC*, pp. 530–533, 1997.
- [11] Y. G. Saab, "An Improved Linear Placement Algorithm Using Node Compaction," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 15, no. 8, pp. 952–958, 1996.
- [12] J. Li, J. Lillis, L.-T. Liu, and C. K. Cheng, "New Spectral Linear Placement and Clustering Approach," in *33rd DAC*, pp. 88–93, 1996.
- [13] S. Sato, "Simulated Quenching: New Placement Method for Module Generation," in *ICCAD*, pp. 538–541, IEEE Computer Society Press, Nov. 1997.
- [14] "ftp://ftp.es.ele.tue.nl/pub/lp.solve."
- [15] S.-W. Hur and J. Lillis, "Relaxation and Clustering in a Local Search Framework: Application to Linear Placement," in *Technical Report UIC-EECS-99-2*, 1999.
- [16] H. Yang and D. F. Wong, "Efficient Network Flow Based Min-Cut Balanced Partitioning," in *ICCAD*, pp. 50–55, IEEE Computer Society Press, Nov. 1994.
- [17] H. Liu and D. F. Wong, "Network Flow Based Multi-Way Partitioning with Area and Pin Constraints," in *ISPD*, pp. 12–17, ACM/IEEE, Apr. 1997.
- [18] C. J. Alpert and A. B. Kahng, "A General Framework for Vertex Orderings, with Applications to Netlist Clustering," in *ICCAD*, pp. 63–69, IEEE Computer Society Press, Nov. 1994.