Constraint Satisfaction in Incremental Placement with Application to Performance Optimization under Power Constraints*

Huan Ren and Shantanu Dutt
Dept. of ECE, University of Illinois-Chicago

Abstract

We present new techniques for explicit constraint satisfaction in the incremental placement process. Our algorithm employs a Lagrangian Relaxation (LR) type approach in the analytical global placement stage to solve the constrained optimization problem. We establish theoretical results that prove the optimality of this stage. In the detailed placement stage, we develop a constraint-monitoring and satisfaction mechanism in a network (n/w) flow based detailed placement framework proposed recently, and empirically show its near-optimality. We establish the effectiveness of our general constraint-satisfaction methods by applying them to the problem of timing-driven optimization under power constraints. We overlay our algorithms on a recently developed unconstrained timing-driven incremental placement method FlowPlace. On a large number of benchmarks with up to 210K cells, our constraint satisfaction algorithms obtain an average timing improvement of 12.4% under a 3% power increase limit (the actual average power increase incurred is only 2.1%), while the original unconstrained method gives an average power increase of 8.4% for a timing improvement of 17.3%. Our techniques thus yield a tradeoff of 75% power improvement to 28% timing deterioration for the given constraint. Our constraint-satisfying incremental placer is also quite fast, e.g., its run time for the 210K-cell circuit ibm18 is only 1541 secs.

1 Introduction

As IC design enters the very deep submicron (VDSM) era, the goal of placement is no longer only minimizing the wire length (WL), though it is a very useful metric for minimizing chip area and improving routability, and has been well tackled by some recent works [14, 15, 16]. Critical path delay and net switching power are also important metrics that will change enormously with different placements. Therefore, recent placement tools need to take several metrics into consideration at the same time. A common approach is focusing on optimizing one metric while trying not to perturb other metrics (constraints) too much. To achieve this, [1, 2] place optimization critical cells first to obtain a better improvement on the targeted metric, and then place other cells according to the constraint metrics; [4, 5, 6] use objective functions that are weighted sum of optimization metrics and constraint metrics. The major disadvantage of these two approaches is that they cannot set an explicit constraint, since it is hard for them to exactly control the constraint-metric change. [3] uses a linear programming (LP) method to optimize switching power and explicitly sets timing constraints in a large number of constraint equations (corresponding to critical and near-critical paths) in the LP formulation. As a result, the solution process is time consuming.

For the multi-objective placement problem, a new design methodology of *targeted incremental placement* has recently been shown to be very promising [7, 8]. Targeted incremental placement starts from an initial placement that has fallen short on some metrics' requirements by certain amounts. It will try to meet the requirements by replacing the cells that are critical to the failed metrics. Compared to performing a completely new placement for the same purpose, the advantage of incremental placement lies in that it will only focus on the parts of the initial placement that are crucial to the optimization metric; this implicitly minimizes the perturbation to other metrics that may already be optimized in the initial placement.

Recent state-of-the-art incremental placement works include [7, 8]. They are both timing-driven placement algorithms, and can achieve a significant critical path delay reduction in a relatively short computation time. While results obtained on the deterioration of other metrics (e.g., net switching power and WL) are acceptable (within 10%) due to the fact that only a small part of the circuit is replaced, none of these two methods explicitly addresses limiting the deterioration of these metrics. However, if the required constraints on other metrics are very tight (e.g., an upper bound deterioration of 3%), then constraint-satisfaction measures must be taken in the incremental placement process. In this paper, we propose constraint-satisfaction techniques for incremental placement that can effectively handle very tight constraints.

Our constraint-satisfaction method uses a Lagrangian Relaxation (LR) type approach in the global placement stage. The LR method is often used in mathematical programming with complex or large number of constraint equations. The relaxation process is obtained by removing some or all the constraints and adding them to the objective function with some coefficients. For example, for the following problem:

$$\text{Minimize } c^T x \text{ subject to}$$
$$Ax \leq b \text{ and } Bx \leq 0$$

If we relax the first set of constraints, the new optimization problem is:

$$\text{Minimize } c^T x + w^T(Ax - b) \text{ subject to}$$
$$Bx \leq 0$$

*This work was supported by NSF grant CCR-0204097.

where w^T is the coefficient for the constraint equations. Choosing different values of w^T will give different relaxed optimization problems. Let $R(w^T)$ be the list of optimal values of the relaxed problems with different w^T, and $x(w^T)$ be the corresponding solutions. The relaxed problem has two properties: 1) $R(w^T)$ is a lower bound on the optimal solution of the original problem with any w^T (a relaxation); 2) If the optimal solution $x(w_0^T)$ to a relaxed problem with $w^T = w_0^T$ satisfies the *complementary slackness condition* $Ax - b = 0$, then it is also the optimal solution to the original problem. The second property is true for the following reason. For any feasible solution x to the original problem other than $x(w_0^T)$, we have $c^T x(w_0^T) + w_0^T (Ax(w_0^T) - b) \leq c^T x + w_0^T (Ax - b)$, and $Ax \leq b$ if x is a feasible solution to the original problem. Thus, since $Ax(w_0^T) - b = 0$, $c^T x(w_0^T) \leq c^T x + w_0^T (Ax - b) \leq c^T x$, which proves that $x(w_0^T)$ is the optimal solution for the original problem. Common LR methods use two steps to solve the original problem, utilizing the two properties. First, we need to find the best lower bound to the original problem, which is $\max\{R(w^T)\}$. Then, we need to test the complementary slackness condition on the corresponding $x(w^T)$ to see if this best lower bound is actually the optimal solution to the original problem.

The key process in an LR approach is finding the coefficient vector w^T that maximizes $R(w^T)$. This is also the major difference between an LR approach and the method of using an objective function of weighted summation of the optimization metric and the constraint metric with a fixed weight, i.e., in LR, the coefficient is determined dynamically for different problem (circuit) instances. It is hard to believe that a fixed weight will suit all circuits.

Our method follows the idea of LR in the global placement stage, and utilizes special properties of incremental placement to speed up the process of finding the best constraint coefficient w^T. Furthermore, to ensure that the constraints are met in the final placement solution, we also construct a constraint-metric monitoring mechanism in the detailed placement stage and prevent detailed placement moves that violate the constraint. In this paper, we focus on the problem of timing-driven incremental placement under dynamic power constraint, though our method is applicable to other *cumulative* constraints, i.e., constraint metrics whose value for a circuit is the sum of their values for relevant circuit components (e.g., interconnects when the metric is dynamic power).

The rest of the paper is organized as follows. Sec. 2 presents the basic problem formulation, while in Sec. 3 we discuss a recent WL model [18] that we use here. In Sec. 4 we present an LR-type method for solving the constrained optimization problem in the global placement stage. In Sec. 5 our constraint satisfaction methods in detailed placement are discussed at length. Section 6 presents experimental results and we conclude in Sec. 7.

2 Problem Formulation

If the initial placement misses the target delay by a certain amount, e.g., 20%, it is possible to meet the timing goal by incremental changes to the placement. Incremental timing-driven placement will seek to improve the timing property by replacing cells on critical and near-critical paths to more "advantageous" positions without significantly impacting the delays on close-to-near-critical paths. As in normal placement, the placement flow usually consists of a global placement stage and a detailed placement stage. In the global placement stage, cells in the critical and near-critical paths are considered *movable* (we denote this set of cells as *moveC*), and their positions are changed to produce shorter paths. The resulting placement often has these movable cells in illegal positions, i.e., either overlapping with other fixed cells or falling between rows (in a standard-cell design). Then, in the detailed placement stage, the adjacent fixed cells are incrementally shifted to empty locations to accommodate the moved cells newly placed into their locations. Finally, a legalized placement is obtained.

In the global placement stage, the problem can be formulated as a mathematical programming problem. In [8], timing improvement is obtained by minimizing the timing objective function:

$$F_t = \sum_{n_j \in moveN} D(n_j)/S_a(n_j) \tag{1}$$

where *moveN* is the set of nets that are connected to *moveC*, $D(n_j)$ is the delay of net n_j, and $S_a(n_j)$ is the slack of n_j. For unrouted nets, $D(n_j)$ can be modeled as in [8]:

$$D(n_j) = R_d(cl(n_j) + Cap_f) + \frac{rc}{2} l_c^2(n_j) + rl_c(n_j) Cap_c \tag{2}$$

where R_d is the driving resistance of the net, c (r) is the capacitance (resistance) per unit WL, Cap_f is the total fan-out capacitance of n_j, l_c is the wire length from the driving cell to the most timing critical sink cell, and Cap_c is the load capacitance of that cell. Minimizing F_t can be solved by quadratic programming as in [8, 13].

If we have a constraint metric C, then a constraint equation must be added:

$$C_o + \Delta C_g + \Delta C_d \leq (1 + \varepsilon)\, C_o \tag{3}$$

where C_o is the value of the constraint-metric function before replacement, ΔC_g and ΔC_d are the constraint-metric changes in global and detailed placement, respectively, and ε is the given fractional upper-bound on the deterioration of the constraint metric. In this paper, we consider minimization metrics, and thus an increase, in either the optimization (e.g., timing) or constraint (e.g., power) metric, is deemed a deterioration. This does not limit the generality of our method, since for the maximization metrics, we can simply take the negation of the metrics. In the global placement stage, the constraint equation can be rewritten as:

$$\Delta C_g \leq \varepsilon\, C_o - \Delta C_d \tag{4}$$

In Eqn. 4, we can see that to satisfy the given constraint, the constraint-metric change in the global stage ΔC_g cannot take up all the allowable deterioration margin $\varepsilon\, C_o$ due to possible constraint-metric deterioration ΔC_d in the detailed stage. Since the actual value of ΔC_d is not available prior to the global stage, we make an estimation of ΔC_d from the deterioration result of C for a tentative purely timing-driven (in general, optimization-metric driven) incremental placement with-

out constraints. In the next section, we give an LR-type approach to efficiently solve the constrained optimization problem in global placement.

In detailed placement, cell movements, either moving fixed cells to make space for movable cells or shifting movable cells into legal positions, will affect delays of some paths. [8] models the effect of shifting a cell u from position p to p' with the following timing cost $T_u(p,p')$:

$$T_u(p,p') = \Delta D_u(p,p') \cdot \frac{1}{S_a(u)^2} \tag{5}$$

where $\Delta D_u(p,p')$ is the delay change of the most critical path through u when cell u is moved from position p to p', and is obtained by differentiating $D(n_j)+D(n_k)$ (Eqn. 2) w.r.t. interconnect length l_c, where n_j, n_k are the two critical-path nets connected to u (see [8] for details); $S_a(u)$ is the slack of nets n_j and n_k connected to u (they will be the same).

For constraint satisfaction, we need to also consider the constraint-metric change associated with each cell movement. A similar constraint-metric cost for each movement is given in Sec. 5. Any cell movement in detailed placement that causes violation to constraints will be temporarily disallowed until the constraint quota increases enough at a later stage (due to other cell movements) to allow the blocked cell movements. Since in global placement, we already leave a margin of ΔC_d for possible constraint-metric deterioration in the detailed stage, the number of timing-optimal movements discarded due to constraint violation has been observed to be very low—less than 4% on the average. Therefore, empirically speaking, our constraint-satisfying detailed placement is near-optimal for the timing metric.

For a dynamic power constraint, the constraint function C is the normalized dynamic power of the circuit:

$$P = \sum_{n_j} c \cdot l(n_j) p_{sw}(n_j) \tag{6}$$

where $l(n_j)$ is the WL of net n_j, and $p_{sw}(n_j)$ is the switching probability of net n_j[1].

3 Using a New WL Model

A popular WL model is the half perimeter bounding box (HPBB) model. This model is accurate for 2 and 3 pin nets, but will underestimate the WL of nets with more pins. Optimizing the HPBB model based WL can be formulated as an LP problem. Most analytical placers use either the clique or the star-graph models for WL estimation of nets, which can be solved faster by quadratic programming than with an HPBB model (which requires an LP formulation). However, the star-graph and clique models are only accurate for 2-pin nets, and tend to overestimate WL when the number of pins is larger than 2.

In this paper, we use the recently proposed *multi-star model* for WL estimation [18]. This model like the standard star-graph model is amenable to quadratic programming formulations, since the coordinates of each cell contribute in a closed-form manner to the WL metric, but with a higher accuracy.

The new model is depicted in Fig. 1. All the pins of a net are divided into sub-groups according to different levels of bounding boxes in a net. Starting with the outermost pins, the groups are formed in the following way. Each time, we first determine the bounding box of all the unselected pins in the net, then we select one pin on each side of the bounding box to form a sub-group of pins. If there are more than one pin on some side of the bounding box, we choose the one that is furthest away from the net centroid (rationale given shortly). Let the first group be cells on the outermost bounding box with the group number increases as we move inwards. For group m, we denote the star graph WL of the group to be l_m. Then, the estimated WL of net n_j is the weighted sum of the star-graph WL of each group, as follows:

$$l(n_j) = \sum_{group\ m \in n_j} \beta^{m-1} l_m \tag{7}$$

where $\beta \leq 1$ is the non-shared fraction of routing between the second and first groups.

The star graph WL l_m of group m is calculated as:

$$l_m = \sum_{u_i \in group\ m} |x_i - x_c| + |y_i - y_c|$$

where (x_c, y_c) is the coordinate of the net center; $x_c(y_c) = (1/k) \times \sum_{u_i \in n_j} x_i(y_i)$. The above non-linear equation can be approximated by quadratic equations as introduced in Gordian-L [13]:

$$l_m = \sum_{u_i \in group\ m} \frac{(x_i - x_c)^2}{|x'_i - x'_c|} + \frac{(y_i - y_c)^2}{|y'_i - y'_c|}$$

where x'_i, x'_c, y'_i and y'_c are the current values of variables x_i, x_c, y_i and y_c.

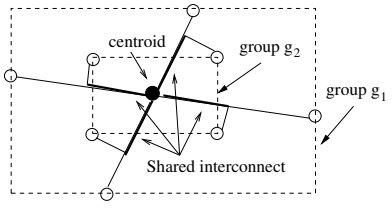

Figure 1: *The multi-star WL model.*

Compared to the standard star-graph WL estimation, which is equivalent to having a weight of one ($\beta = 1$) for every group to determine the total WL of the net, Eqn. 7 assigns exponentially smaller weights (β^{m-1}) for groups that are closer to the center. This is based on the observation that, the interconnects belonging to those groups often share significant parts with interconnects of outer groups as shown in Fig. 1. It is due to this reason that if there are more than one node on any side of a certain level

[1]The dynamic power dissipation P_{n_j} of a net n_j is $P_{n_j} = 0.5V^2 C f p_{sw}$, where V is the supply voltage, C is the total load capacitance of the net, and f is the clock frequency of the circuit. The total load capacitance consists of two components: the sink gate load C_{gate} which is the sum of total input capacitance of fanout gates of the net and the drain to ground capacitance of the driver; the wire load C_{wire} which is $cl(n_j)$. In the placement stage, V, f and C_{gate} are fixed. Therefore, for simplicity we can omit these constant terms in the power constraint function.

Circuit	matrix	vp2	mac32	mac64	%error
routed WL	1.2	4.2	4.8	26.6	0
star-graph	1.8	5.6	5.8	35.0	35.5
HPBB	1.0	3.4	4.0	20.4	-19.0
multi-star	1.2	4.1	5.5	29.0	6.6

Table 1: *The estimated WL using the three different WL models. The unit here is $10^5 \mu m$. The actual routed WL is also listed for comparison.*

of bounding box, we choose the one that is furthest away from the net centroid, since the other nodes on that side tend to share common interconnects with the furthest one. The value of β is determined by the degree of overlapping among interconnects of cells across all levels of adjacent bounding boxes. Experiments reveal that choosing $\beta = 1/3$ consistently gives the most accurate routed WL prediction [18].

Table 1 shows total WL estimations using the multi-star model, the standard star-graph model and the HPBB model for the TD-Dragon benchmarks, and their percentage differences (%error) from the routed WL [18]. As shown by the results, the multi-star model provides a significantly better approximation of the routed WL than the standard star-graph and HPBB models.

4 Constraint Satisfaction in Global Placement

Constraint consideration in the global placement stage is necessary to provide the detailed placer with a raw layout that is possible to legalize while satisfying the given constraints.

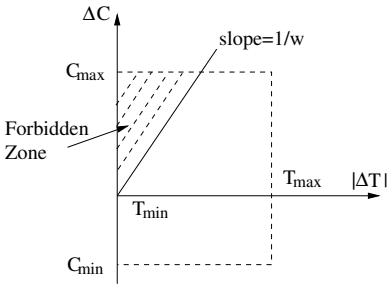

Figure 2: *The 2-D plane of changes of power and timing metrics due to cell movements in global placement.*

For timing-driven incremental placement under constraints, the objective function F_t is the same as in purely timing-driven placement (Eqn. 1), while the constraint expression is given in Eqn. 4. An LR approach is helpful in solving these types of optimization problems that have complex constraint equations. After relaxation, we will get a new unconstrained optimization problem with the objective function:

$$F_r = F_t + w(\Delta C_g - (\varepsilon\, C_o - \Delta C_d)) \tag{8}$$

As we discussed in Sec. 2, C_o and ΔC_d are the original values of the constraint metric C and its allowable change in detailed placement, respectively, that are constants in the global placement stage, F_t is the timing objective function, and ΔC_g is the constraint metric change in global placement.

The determination of coefficient w is critical. In an LR formulation such as Eqn. 8, to get the same optimal solution as the original constrained problem, we have to choose w so that the optimal solution of Eqn. 8 is maximized among all optimal solutions for different w's (see Sec. 1 for more details). To efficiently obtain this w value, we need to first determine the relationship between the change of the timing objective function F_t and of the constraint metric C w.r.t. w.

Movement of a cell v in global placement will cause changes in the values of both F_t and C. To depict the effect of each movement, we can construct a two dimensional plane, in which the y coordinate is the constraint-metric change, and the x coordinate is the absolute value of the timing objective function change (in timing-driven global placement, all cell movements have a non-positive delay change, and so we use its absolute value); see Fig. 2.

Cell movements are located in the plane according to the corresponding constraint metric and timing objective function change. All movements fall in the *movement region* bounded by (T_{min}, T_{max}) in the horizontal direction and (C_{min}, C_{max}) in the vertical direction, as shown in Fig. 2, where $T_{min}, T_{max}, C_{min}, C_{max}$ are the minimum and maximum changes of timing and constraint metrics, respectively, across all cell movements.

In a purely timing-driven run (i.e., without any constraints), all possible movements that result in an improved timing will be taken. However, when we use the LR objective function F_r, some movements that have relatively small $|\Delta F_t|$ (timing improvement), but large ΔC (constraint metric deterioration) will not be taken. It should be noted that $|\Delta F_t|$ and ΔC caused by the movement of a cell is dependent on the positions of its adjacent cells. In order to accurately determine which movements should be taken in the relaxed problem, $|\Delta F_t|$ and ΔC of moving a cell should be calculated with all the other movable cells at their optimal position w.r.t. the relaxed problem. Thus, for a movement M_u of cell u taken in the purely timing-driven run, if its $\frac{\Delta C}{|\Delta F_t|}$ is greater than $1/w$, M_u will not be taken for the relaxed problem. This is because the change of F_r corresponding to these movements is $w\Delta C_g - |\Delta F_t| > 0$; therefore, comparing two placement results with all the other movable cells at their optimal positions, the one with M_u will be inferior w.r.t. F_r to the one that keeps u at its original position. Such undesired movements for the relaxed problem fall into the triangular shaded area in Fig. 2, which is termed the *forbidden zone*. The slope of the boundary line of the shaded region is $1/w$ (obtained by setting $w\Delta C_g - |\Delta F_t| = 0$, and taking the ratio of $\frac{\Delta C}{|\Delta F_t|} = 1/w$).

If we know the values F_t^T and ΔC_g^T of F_t and ΔC_g, respectively, in a timing-optimal solution with no constraints, then the optimal solution of the relaxed objective function F_r can be obtained by removing from F_t^T and ΔC_g^T the changes in F_t and C_g caused by movements in the forbidden region, since these movements will not be taken for minimizing F_r. Therefore, the optimal value $F_r^{opt}(w)$ of F_r is:

$$F_r^{opt}(w) = (F_t^T - \Delta F_t^{fb}(w)) + w((\Delta C_g^T - \Delta C^{fb}(w)) - (\varepsilon\, C_o - \Delta C_d)) \tag{9}$$

where $\Delta C_g^{fb}(w)$ is the total constraint metric increase due to the movements in the forbidden zone, and $\Delta F_t^{fb}(w)$ is the total improvement (reduction) of the timing objective function F_t obtained from the movements in the forbidden zone. Note that in the above equation, $F_t = F_t^T - \Delta F_t^{fb}(w)$ and $C_g = \Delta C_g^T -$

$\Delta C^{fb}(w)$.

To obtain the best lower bound on the optimal solution of the original problem F_t, we need to solve another optimization problem: $\max\{F_r^{opt}(w)\}$, which is tackled by setting the differentiation of Eqn. 9 w.r.t. w to be 0. The differentiation of Eqn. 9 can be obtained easily by utilizing a relationship between $\frac{d\Delta F_t^{fb}(w)}{dw}$ and $d\frac{\Delta C^{fb}(w)}{dw}$. When w is increased by a small amount dw, the forbidden region is enlarged, and movements whose $\frac{|\Delta F_t|}{\Delta C}$ is between w and $w+dw$ is added to the forbidden region. Since dw is small, for these movements $|\Delta F_t| \approx w\Delta C$. The change $d\Delta F_t^{fb}(w)$ of $\Delta F_t^{fb}(w)$ when w is increased to $w+dw$ is the summation of $|\Delta F_t|$ of these movements; similarly, $d\Delta C_g^{fb}(w)$ is the summation of ΔC of these movements. Therefore, we have $d\Delta F_t^{fb}(w) = w d\Delta C_g^{fb}(w)$, which implies:

$$\frac{d\Delta F_t^{fb}(w)}{dw} = w\frac{d\Delta C^{fb}(w)}{dw} \tag{10}$$

Substituting Eqn. 10 into the differentiation of Eqn. 9, we can derive the equation for solving w:

$$(\Delta C_g^T - \Delta C^{fb}(w)) - (\varepsilon\, C_o - \Delta C_d) = 0 \tag{11}$$

The second-order differentiation of Eqn. 9 is $-d\Delta C^{fb}(w)/w$. Since increasing w will also increase the size of the forbidden region (note the slope of the boundary of the forbidden region is $1/w$), $\Delta C^{fb}(w)$ will increase accordingly. Thus, $d\Delta C^{fb}(w)/w > 0$, and $-d\Delta C^{fb}(w)/w < 0$. Therefore, w derived from Eqn. 11 gives the maximum $F_r^{opt}(w)$. After deriving w, we can solve the relaxed problem (Eqn. 8). However, we need to test the complementary slackness condition on the solution to the relaxed problem to see if it is also the optimal solution to the original problem (see Sec. 1). It is interesting to note that the complementary slackness condition $\Delta C_g - (\varepsilon\, C_o - \Delta C_d) = 0$ is the same as Eqn. 11, since $\Delta C_g = \Delta C_g^T - \Delta C^{fb}(w)$. Therefore, with w obtained from Eqn. 11, the complementary slackness condition is guaranteed to be met, which means that the optimal solution of the relaxed objective function F_r with w derived from Eqn. 11 is also the optimal solution to the original problem (the basis of this conclusion is explained in the description of the general LR approach in Sec. 1).

With the above analysis, we establish the following theorems.

Theorem 1 *If we set the coefficient $w = w_o$ as derived from Eqn. 11, then the constraint will be met.*

Proof: Since the complementary slackness condition $\Delta C_g - (\varepsilon\, C_o - \Delta C_d) = 0$ is met when $w = w_o$, the total constraint-metric change after incremental placement $\Delta C_g + \Delta C_d$ equals the upper bound deterioration $\varepsilon\, C_o$ of constraint C. Therefore, the theorem is true. ♠

Theorem 2 *If we set the coefficient $w = w_o$ as derived from Eqn. 11, then it can give the optimal delay objective function value while satisfying the given constraint.*

Proof: For any other coefficient value w', if w' is smaller than w_o, then the forbidden region will reduce (note the slope of the boundary of the forbidden region is $1/w$), which means a decrease in $\Delta C^{fb}(w)$, and thus an increase in C_g. According to Theorem 1, the total constraint-metric deterioration with $w = w_o$ is equal to the upper bound constraint $\varepsilon\, C_o$. Thus, with a smaller w', the constraint is not satisfied. If w' is larger than w_o, then the forbidden area is enlarged and more timing improvement movements are forbidden, which gives a inferior delay objective function value compared to w_o. ♠

The above theorems and analysis hold for general constraint metrics. All that is needed is to obtain the closed-form expression for $\Delta F_t^{fb}(w)$ and $\Delta C^{fb}(w)$, solve Eqn. 11 to obtain w_o, and then perform analytical global placement for optimizing F_r (Eqn. 8) using $w = w_o$.

Application to timing optimization under power constraint: For the dynamic power constraint, the constraint metric C is the net switching power P. We relabel the parameters related to C, ΔC_g^T, C_o, ΔC_d, C_{max} and C_{min} as ΔP_g^T, P_o, ΔP_d, P_{max} and P_{min}, respectively. It is observed from experiments that all cell movements are roughly uniformly distributed in the two dimensional space of Fig. 2 for most circuits in the 27 benchmarks we use. The possible reason for this is that though both net switching power and net delay depend on the WL of a net, the coefficients of the WL are different for the power and delay functions, and are independent of each other. Suppose the WL of a net n_j is changed by Δl due to a cell movement. Recalling Eqns. 1 and 6, the power change ΔP is $c\Delta l\, p_{sw}(n_j)$, where $p_{sw}(n_j)$ is the switching probability of n_j, and c is the unit wire capacitance; the timing objective function change ΔF_t is $\frac{R_d c}{S_a(n_j)}\Delta l$, where R_d is the driving resistance of the net, and $S_a(n_j)$ is the slack of n_j. It is easy to see that the two terms p_{sw} and $\frac{R_d}{S_a(n_j)}$ are independent of each other, implying the independence between the two coefficients $c\,p_{sw}(n_j)$ and $\frac{R_d c}{S_a(n_j)}$.

Furthermore, it is reasonable to assume a uniform distribution of p_{sw}. In our experiments, we have tried two different assignments of p_{sw}, random assignment (i.e., uniform distribution) and an assignment that sets p_{sw} of a net as a linear function of the largest distance of the net from any input in order to model the increasing number of glitches on nets that are further from circuit inputs. The results of these two assignments are virtually the same for all circuits. Thus, assuming a uniform distribution of p_{sw} is a good approximation to the actual p_{sw} distribution. For the $\frac{R_d}{S_a(n_j)}$ term in ΔF_t, since the movable cells in timing-driven placement are cells on critical or near-critical paths, the slacks $S_a(n_j)$ are alike for all nets on these paths, while R_d is usually uniformly distributed; this implies that $\frac{R_d}{S_a(n_j)}$ is uniformly distributed across all nets. Therefore the movement of a cell can result in any metric change point $(\Delta F_t, \Delta P)$ in the 2-D space of Fig. 2 with equal probability. Thus, the resulting distribution of movements in the above space is roughly uniform.

Because of the uniform distribution, the number of movements $m = |moveC|$ (note that $|moveC|$ is determined a-priori, where recall from Sec. 2 that $moveC$ is the set of movable cells in the global placement) that fall in a unit area in the movement region of the two dimensional space of Fig. 2 is m/A, where $A = (T_{max} - T_{min}) \times (P_{max} - P_{min})$ is the area of

the movement region in the 2D-space shown in Fig. 2. Thus, the density of movement in the movement region is m/A. For a small region $dxdy$ at (x,y) in the movement region, there are on the average $(m/A)dxdy$ movements with a constraint metric change of y and a delay metric change of x. Therefore, the total power increase $\Delta P^{fb}(w)$ due to movements in the forbidden zone thus is:

$$\Delta P^{fb}(w) = \int\int_{forb.\ zn.} y(m/A)\ dxdy = (m/A)\int_{y=0}^{P_{max}}\int_{x=0}^{wy} y\ dxdy \tag{12}$$

Similarly, the total improvement of timing objective function F_t given by the movements in the forbidden zone is:

$$\Delta F_t^{fb}(w) = (m/A)\int_{y=0}^{P_{max}}\int_{x=0}^{wy} x\ dxdy \tag{13}$$

Using Eqns. 12 and 13 in Eqn. 11 and solving it, we get:

$$w_o = 3A(\Delta P_g^T - (\varepsilon\, P_o - \Delta P_d))/(mP_{max}^3) \tag{14}$$

The parameters needed for calculating the value of A are T_{min}, T_{max}, P_{max} and P_{min}. It should be noted that the accurate value of these four boundary parameters are unknown before we actually solve the relaxed problem, since by definition, we need the optimal positions of cells for the relaxed problem to determine ΔF_t and ΔP for each movement. However, we can make an estimation of the boundary values using the cell position after the global placement stage of a purely timing-driven run. Experiments show that the variation of the estimated values from the actual values obtained after solving the relaxed problem is within 5% for the benchmarks we use, which is acceptable. However, if the difference is large, we can use an iterative way to obtain accurate values as follows: (1) calculate w_o using the estimated value; (2) then solve the relaxed problem, and obtain the optimal cell positions; (3) after that, recalculate the four parameters using the optimal positions; (4) finally, recalculate w_o, and repeat step (2). Since solving the global placement problem is fast (recalling that we are using quadratic programming), we can get very accurate values in reasonable run time.

From the tentative purely timing-driven run, we can also obtain the values of other parameters for Eqn. 14: ΔP_g^T, P_o and ΔP_d. If the resulting placement from this tentative run itself already satisfies the constraint, no more processing is needed; otherwise we use these parameters to determine coefficient w_o, and perform incremental global placement by using the relaxed objective function F_r (Eqn. 8) with $w = w_o$. Since for optimizing the objective function F_r, the movements in the forbidden zone will no longer be taken, we cannot directly use the power change value ΔP_d^T of the purely timing-driven run as the ΔP_d value. ΔP_d should be smaller than ΔP_d^T, because fewer movements are taken when optimizing F_r than when optimizing F_t without any constraint. Thus, it is reasonable to set ΔP_d to be:

$$\Delta P_d = \Delta P_d^T \cdot (1 - \frac{A_{fb}}{A}) \tag{15}$$

where $A_{fb} = (wP_{max}^2)/2$ is the area of the forbidden region.

The above discussion and Theorems 1 and 2 lead to the following result.

Theorem 3 *For the problem of timing optimization under dynamic power constraint, if we set the coefficient $w = w_o$ as that determined in Eqn. 14, then optimizing F_r (Eqn. 8) with $w = w_o$ gives the optimal value of the delay objective function F_t (Eqn. 1) while satisfying the power increase upper-bound constraint ΔP_g.*

5 Constraint Satisfaction in Detailed Placement

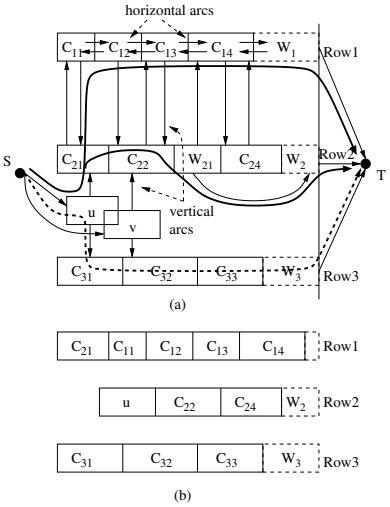

Figure 3: *(a) General n/w flow graph and flows for legalizing placement of cell u. C_{ij}'s are the fixed row cells, u and v are the moved cells placed in global placement, W_{ij}'s are the available white space (WS). Two alternative legalizing flows are shown in solid and dashed lines respectively. (b) The resulting position corresponding to the solid flows in (a), which legalize u. White spaces in Row* 1 *and Row* 2 *are decreased. Similarly, following the alternative dashed flow will take up the white space of Row* 3.

The technique in [8] models detailed placement as a min-cost network (n/w) flow problem. Figure 3 shows the general structure of the n/w graph. The vertical arcs model the movement of their start cells into adjacent rows, and the capacity of each such arc equals the start cell width. The horizontal arcs model the horizontal movement of their start cell in the arc direction, and their capacities are the maximum distances the start cells can move horizontally. This maximum distance for row R_i is determined as the maximum of the widths of the illegally placed cells that can enter R_i and the widths of cells in R_{i+1} and R_{i-1}. This finite capacity is chosen for horizontal arcs in R_i in order to balance cost accuracy and flexibility in cell movement. The timing cost of an arc is calculated using the function $T_u(p,p')$ described in Eqn. 5, divided by the capacity of the arc, where u is the start cell of the arc, p is the original position of u and p' is the new position of u when a full flow passes through the arc (e.g., for vertical arcs, p' is the position located in the adjacent row with the same horizontal coordinate as p; for horizontal arcs, p' is at the maximum distance a cell can move from p in the same row). The source (S) has arcs to the moved cells that were placed in global placement, and are in illegal positions. Available white spaces in each row are connected to the sink (T). Thus, as shown in Fig. 3 when we send a flow from S to T via the illegally-placed cells, we are moving them to their adjacent rows and creating space for these new incoming cells by shifting other cells in the rows towards available white spaces. Therefore, performing a

min-cost flow achieves placement legalization with minimum deterioration of the objective function. The advantage of this method is solving min-cost legalization problem with a time efficient continuous optimization method (n/w flow), despite the fact that this problem is an integer programming problem; see [8] for further discussion on these issues.

The min-cost n/w problem is solved using the Simplex method [9]. The Simplex method starts from an initial non-optimal flow. It keeps determining *negative cost cycles* in the n/w flow graph and augmenting flows in these cycles, thereby improving the total cost. The process stops when there are no more negative cost cycles or no more flow can be augmented in any negative cycle (due to one or more arcs in the negative cycle being saturated in the direction of the cycle or empty in the opposite direction). In the classic Simplex method that we use, negative cycles are augmented in order decreasing cost improvement (i.e., cycles with larger negative cost magnitudes are augmented before cycles with smaller negative cost magnitudes).

For a purely timing-driven detailed placement, we only need to consider the timing cost of each arc. When a constraint is added, we need also to know how much the constraint metric is changed when augmenting flows. Therefore, a similar constraint metric cost is calculated for each arc, which equals the constraint-metric change when a cell is moved according to this arc. For dynamic power, if a cell u is moved from position p to p' among an arc a_k, the cost of this movement is:

$$\Delta P_u(p,p') = \sum_{u \in n_j} c \cdot (l_{p'}(n_j) - l_p(n_j)) p_{sw}(n_j)$$

Recall that c is the unit length wire capacitance, $p_{sw}(n_j)$ is the switching probability of net n_j, and $l_p(n_j)$ is the WL of net j with cell u at position p. The above equation gives the total power change of all the connected nets to cell u when cell u is moved from p to p', since only these nets will be affected by the movement of cell u. Then the unit flow power cost of the arc a_k is $\Delta P_u(p,p')/cap(a_k)$.

We thus have two costs for each arc for timing optimization under power constraints (in general, we can have multiple costs, one for the optimization metric and the others for multiple constraint metrics). Thus, for each negative timing cost cycle, we can also calculate the unit flow constraint metric cost of the cycle. To meet the constraint, the total constraint metric cost (deterioration) should be no larger than the available margin after global placement $\varepsilon C_o - \Delta C_g$.

If we find that augmenting a single unit flow in one negative cycle will cause a constraint violation, we disallow flow augmentation in this cycle by putting it into a *forbidden list*. However, when this situation is not true anymore (e.g., some subsequent flows cause a decrease of the constraint metric), we free cycles from the list that will not causes a constraint violation due to a unit flow through them. This constraint-satisfaction technique is essentially a greedy method, i.e., we find the cycle with largest negative timing cost, check if augmenting flow in the cycle will cause constraint violation, augment flow in the cycle if not, then proceed to the next cycle with the next largest negative timing cost and so forth. Since obtaining the timing-optimal flow under constraints of total cost of other metrics in a network flow graph is typically a non-convex problem, solving it with a greedy method is sub-optimal. However, our experiments with power as the constraint show that among all cycles with negative timing cost, only about 4% are disallowed in our technique due to violation of the power constraint. This small number is due to the reasonable power deterioration margin ΔP_d we leave for detailed placement in the global placement stage (see Eqn. 15). Thus from the above empirical evidence, we can conclude that our constraint-satisfaction technique in detailed placement is near-optimal.

6 Experimental Results

We use three benchmark suites in our experiments: 1) the TD-Dragon suite of [5], 2) Faraday benchmarks from [10] and 3) TD versions of the IBM benchmark suite of [8, 12]. All benchmarks are initially placed by Dragon with WL as the objective metric. The switching probability p_{sw} of each net is assigned between 0.1 and 1 as a linear function of the longest distance of the net from an input; besides the current to new value switching, this is an attempt to model net switching power due to glitches which increase as the distance of a net from an input. Similar results were obtained for a uniform distribution of p_{sw} between 0.1 and 1 among all the nets. Note that these values of p_{sw} are relative, not absolute, estimations of switching activity; thus, for example, if the actual switching probability of each net is 1/10 of our p_{sw} assignments, the results in terms of percentage power deterioration will be exactly the same. We ran our programs on Linux and Windows XP Pentium IV machines with up to 1GB of main memory, and almost the same program execution speeds.

In Table 2, we give the characteristics of the benchmark circuits as well as the purely timing-driven incremental placement results. The average power increase after purely timing-driven placement is about 8.7%, while the average delay improvement is 17.3%. Table 3 shows the results of constrained timing-driven incremental placement with a power constraint of 3% (upper bound of net switching power increase is 3%). It can be seen that the constraint is met for nearly all the benchmarks with an actual average power increase of only 2.1% (a 75% relative decrease in power deterioration compared to the unconstrained case) and a resulting average timing improvement of 12.4% (a 28% relative decrease in timing improvement compared to the unconstrained case). Only one circuit misses the constraint by 0.1%, probably due to the small inaccuracy in the power cost calculated in the n/w flow based detailed placer.

To show the effectiveness of our LR-type global placement method, we also give results in Table 3 for coefficient values w in the relaxed objective function of Eqn. 8 that are slightly different from the optimal value w_o (Eqn. 14); the values chosen are $1.1w_o$ and $0.9w_o$. Timing results for w_o are consistently better than those of the slightly changed coefficients with the exception of only two circuits, for which the $0.9w_o$ coefficient is no more than 0.3% better. However, the $0.9w_o$ coefficient choice fails to meet the constraint for 4 circuits. The efficacy of our detailed placer is validated by small deteriorations in the timing results when going from global to

Ckt	# of cells	Power (10^{-8} mw)	Init. delay (ns)	Init. pl runtime (secs)	%ΔT (global)	%ΔT	%ΔP	CPU (secs)
ibm01	12.5K	1.3	2.2	402	21.0	15.0	-10.5	78
ibm02	19.3K	5.8	1.8	864	30.5	24.2	-11.4	76
ibm03	22.8K	6.0	1.2	1283	32.3	28.5	-9.4	169
ibm04	27.2K	10.2	1.7	1501	28.1	24.1	-11.6	177
ibm05	28.1K	14.1	1.1	1594	21.2	17.9	-6.9	182
ibm06	32.3K	9.1	2.6	1800	25.9	21.0	-9.1	223
ibm07	45.6K	14.0	2.3	2216	18.0	13.1	-8.0	249
ibm08	51.0K	14.3	1.1	5973	29.2	27.1	-6.2	248
ibm09	53.1K	18.6	3.0	4032	17.4	13.3	-10.7	349
ibm10	68.7K	17.0	2.2	4578	20.2	16.2	-9.9	352
ibm11	70.2K	13.2	2.0	4415	21.2	14.4	-9.4	431
ibm12	70.4K	13.5	5.4	4850	19.1	13.2	-6.9	301
ibm13	83.7K	22.4	1.8	5189	22.1	18.3	-7.1	312
ibm14	147K	53.2	3.0	7432	19.5	14.8	-8.5	469
ibm15	161K	63.6	3.8	7629	22.6	18.4	-3.8	488
ibm16	183K	62.0	3.8	7714	24.6	18.5	-4.4	596
ibm17	185K	59.6	4.5	8259	30.7	27.1	-2.4	684
ibm18	210K	103.4	2.0	9454	36.2	33.4	-3.7	762
Avg.				**4399**	**24.5**	**19.9**	**-7.8**	**342**
DMA	11.7K	1.4	0.4	384	25.1	14.4	-13.8	58
DSP1	26.3K	1.4	0.9	1527	24.0	16.1	-10.8	72
DSP2	26.3K	1.4	0.8	1602	23.0	15.0	-9.2	90
RISC1	32.6K	7.4	1.1	1952	21.8	16.2	-9.0	108
RISC2	32.6K	7.9	1.3	1906	19.9	15.0	-9.3	116
Avg.				**1474**	**22.5**	**15.4**	**-10.4**	**89**
matrix	3.1K	2.8	4.9	70	9.7	7.1	-9.6	80
vp2	8.7K	4.4	5.1	161	10.8	6.0	-9.9	110
mac32	8.9K	7.1	3.8	184	13.7	9.3	-8.1	102
mac64	25.6K	18.5	7.7	1364	13.1	10.2	-6.9	135
Avg.				**445**	**11.6**	**8.1**	**-8.7**	**107**
Overall Avg.				**3271**	**22.3**	**17.3**	**-8.4**	**260**

Table 2: *Results for timing-driven incremental placement without constraints. All circuits are initially placed by Dragon, a WL-optimizing placer. The size, initial power and delay of each benchmark are given, followed by the % change of delay after incremental global placement (%ΔT(global)), and final % changes in delay (%ΔT) and power (%ΔP) after incremental detailed placement.*

Ckt	%ΔT (global)	%ΔT	%ΔP	runtime (secs)	%ΔT ($0.9w_o$)	%ΔT ($1.1w_o$)
ibm01	15.8	11.3	-2.9	277	10.4	11.0
ibm02	19.4	16.0	-2.8	331	15.9	15.6
ibm03	20.7	18.0	-1.4	388	17.9	17.5
ibm04	17.0	14.7	-3.1	612	Fail	13.8
ibm05	14.6	11.2	-3.0	594	10.1	10.3
ibm06	17.4	14.8	-1.4	528	14.9	14.5
ibm07	11.5	8.9	-1.6	714	8.9	8.4
ibm08	28.5	25.1	-2.5	877	24.5	24.6
ibm09	12.3	8.3	-1.8	902	7.3	7.5
ibm10	13.4	10.7	-2.9	1148	Fail	9.4
ibm11	15.3	10.1	-2.9	1024	9.2	10.0
ibm12	13.9	9.8	-2.5	953	9.7	9.7
ibm13	17.5	12.9	-1.4	833	12.9	12.1
ibm14	10.3	8.4	-0.7	1071	9.4	8.1
ibm15	19.6	15.6	-2.9	1248	Fail	14.8
ibm16	19.7	17.0	-2.4	1306	16.6	16.4
ibm17	29.7	27.8	-2.4	684	27.8	27.8
ibm18	27.5	25.0	-2.5	1541	23.8	24.1
Avg.	**18.0**	**14.7**	**-2.3**	**835**	**12.2**	**13.7**
DMA	9.1	5.1	-2.7	414	Fail	4.2
DSP1	16.4	12.2	-2.2	415	11.8	11.2
DSP2	16.5	13.9	-3.0	712	10.9	13.0
RISC1	14.8	10.6	-2.3	538	10.7	9.7
RISC2	15.1	9.9	-2.0	538	9.3	9.3
Avg.	**14.4**	**10.4**	**-2.5**	**515**	**8.6**	**9.5**
matrix	7.4	3.4	-3.0	482	1.6	2.5
vp2	4.8	1.6	-1.2	286	1.6	1.4
mac32	9.6	5.8	-2.5	476	4.9	4.9
mac64	10.2	7.5	-1.1	573	7.5	7.4
Avg.	**7.9**	**4.6**	**-2.0**	**454**	**4.0**	**4.1**
Overall Avg.	**15.8**	**12.4**	**-2.1**	**719**	**10.3**	**11.5**

Table 3: *Results of timing-driven incremental placement under a power constraint of 3% deterioration. The last two columns show results for constraint-metric coefficients in the relaxed objective function that are slightly different from the theoretical optimal choice w_o. "Fail" means that the final placement did not satisfy the given constraint.*

detailed placement—an absolute change of 5% for the unconstrained problem (Table 2) and an absolute change of 3.4% for the power-constrained problem (Table 3).

Since in power-constrained timing-driven incremental placement, we need a test run of purely timing-driven placement to obtain some characteristic parameters about each benchmark, our average run time is about 2.8 times that of purely timing-driven placement. However, our average run time for the power-constrained timing-driven problem is still only 22% of the average run time of the Dragon placer.

7 Conclusions

We presented a novel and efficient algorithm for solving the constrained incremental placement problem. We employ an LR-type method, and make use of the relationship between the optimization metric and the constraint metric to efficiently solve the optimal constraint-metric coefficient selection problem. We also prove the optimality of our constraint-satisfying global placer. In our network-flow based detailed placer, the constraint metric change is dynamically monitored, and movements that cause constraint violation are temporarily disallowed until the constraint-metric deterioration quota increases sufficiently. We applied our general methodology to the power-constrained timing-optimization problem. Results show that: (a) our method can satisfy explicit and tight constraints without significantly sacrificing the optimization metric, and (b) our choice of the constraint-metric coefficient provides the best improvement for the optimization metric, thus empirically validating our theoretical results.

References

[1] N. Togawa, K. Ukai, M. Yanagisawa and T. Ohtsuki, "A Simultaneous Placement and Global Routing Algorithm for FPGAs with Power Optimization", *Proc. IEEE Asia-Pacific Conf. Circuits and Systems*, 1998, pp. 125-128.

[2] A. Kahng, S. Mantik and I. Markov, "Min-Max placement for large scale timing optimization", *ISPD'02*, 2002, pp. 143-148.

[3] H. Vaishnav and M. Pedram, "PCUBE: A Performance Driven Placement Algorithm for Low Power Designs", *Proc. DAC with EURO-VHDL*, 1993, pp. 72-77.

[4] Y.Cheon, P. Ho, A. Kahng, S. Reda and Q. Wang "Power-Aware Placement", *Proc. DAC*, 2005, pp. 795-800.

[5] Y. Xiaojian, C. Bo-Kyung, M. Sarrafzadeh, "Timing-driven placement using design hierarchy guided constraint generation", *ICCAD*, 2002, pp. 177-180.

[6] H. Vaishnav and M. Pedram. "Delay optimal partitioning targeting low power VLSI circuits", *ICCAD*, November 1995, pp. 638-643.

[7] C. Wonjoon, K. Bazargan, "Incremental placement for timing optimization", *ICCAD*, 2003, pp. 463-466.

[8] S. Dutt, H. Ren, F. Yuan and V. Suthar, "A Network-Flow Approach to Timing-Driven Incremental Placement for ASICs", *ICCAD 06*, pp. 375-382.

[9] R. Ahuja, et al., "A Network Simplex Algorithm with O(n) Consecutive Degenerate Pivots", *Operations Research Letters*, 1995, pp. 1417-1436.

[10] S. N. Adya, et al., "Unification of Partitioning, Floorplanning and Placement", *ICCAD*, 2004, pp. 41-46.

[11] S. N. Adya, I. L. Markov, "Consistent Placement of Macro-Blocks using Floorplanning and Standard-Cell Placement", *ISPD*, April 2002, pp. 12-17.

[12] The FlowPlace page: http://www.ece.uic.edu/~dutt/benchmarksetc/FlowPlace/flow.html.

[13] G. Sigl, K. Doll and F. Johannes. "Analytical placement: A linear or a quadratic objective function?", *Proc. ACM/IEEE DAC*, 1991, pp. 427-432.

[14] T. F. Chan, J. Cong and K. Sze, "Multilevel Generalized Force-directed Method for Circuit Placement," *ISPD*, 2005, pp. 185C192.

[15] A. B. Kahng, and Q. Wang, "Implementation and Extensibility of an Analytic Placer", *ISPD*, April 2004, pp. 18-25.

[16] M. Pan, N. Viswanathan and C. Chu, "An Efficient and Effective Detailed Placement Algorithm". *ICCAD*, 2005, pp. 48-55.

[17] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, "Nonlinear Programming: Theory and Algorithms", 2nd ed. New York: Wiley, 1993.

[18] S. Dutt, et al., "Timing-Driven Incremental Placement with Wire Length Considerations using Analytical and Network-Flow Techniques", submitted journal paper.