Extending Pitchmatching Algorithms to Layouts with Multiple Grid Constraints

Hiroshi Miyashita

NTT LSI Laboratories
Atsugi-shi, Kanagawa, 243-01 Japan
Tel: +81 462 40 2136
Fax: +81 462 40 2162
e-mail: miyashit@aecl.ntt.jp

Abstract— Pitchmatching algorithms are widely used in layout environments where no grid constraints are imposed. However, realistic layouts include multiple grid constraints which facilitate the applications of automatic routing. Hence, pitchmatching algorithms should be extended to those realistic layouts. This paper formulates a pitchmatching problem with multiple grid constraints. An algorithm for solving this problem is constructed by extending conventional pitchmatching algorithms. The computational complexity is also discussed in comparison with a conventional naive algorithm. Finally, examples and application results to realistic layouts are presented.

I. Introduction

Compaction-based layout approaches have been widely used in designing leaf cells [3], macro cells [6], and entire chips [2]. Compactors can sqeeze layouts into the smallest possible areas without violating design rules. These approaches have two advantages over conventional manual layouts: First, they can adapt already-designed layouts to the new design rules. In realistic design environments, design rules often change. Since compaction methods enable designers to reuse already-designed layouts, design period can be reduced. Second, they can be applied to module assembly techniques that can automatically generate macro cell layouts from the constituent leaf cell layouts [6]. This enables layout designers to automatically adjust the ports positions of each leaf cell layout so as to construct a macro cell layout even if the layouts of certain leaf cells change.

The first advantage offers a solution to the fundamental problem solved in VLSI layout because the observance of design rules is indispensable in the layout designs. Initially, compaction algorithms could minimize only layout areas taking into account minimum or maximun constraints specified in design rules or by users [8]. Next, they were improved so that they could be applied to realistic problems in VLSI layout designs. For example, the conventional compaction methods have been extended in order to create an algorithm for minimizing wiring lengths [11]. That algorithm can prevent degradations in circuit performance caused by an unnecessary increase in parasitics capacitances. Recently, those methods have been extended to layouts having grid constraints. The grid constraints restrict the coordinates of layout elements to the form $c_{grid} \times n + c_{delta}$ for some integer n given $c_{grid} > 0$ and c_{delta} [4]. However, realistic layouts usually have different kinds of grid constraints; for example, $c_{1,grid} \times n + c_{1,delta}$ and $c_{2,grid} \times n + c_{2,delta}$. From here on, we call this kind of constraints multiple grid constraints. The extension of compaction algorithms to layouts having grid constraints with only a single kind of grid constant is discussed in [4]. The same author also presents compaction algorithms for layouts having multiple grid constraints [5]. Similar kinds of problems are considered in [7].

When compaction is applied to the layout design of macro cells, the second advantage, the layout is constructed by combining constituent leaf cell layouts so that the necessary connectivities between adjacent leaf cell ports can be maintained (See Fig. 1). Thus, each leaf cell must be made as small as possible under the condition that the coordinates of those port-pairs become equal. This process is called pitchmatching. Until now, pitchmatching algorithms have been proposed in many papers [1],[6]. These algorithms enable designers to reuse leaf cell layouts in a macro cell layout configuration. However, all of them are limited to layouts with no grid constraints. Thus, they cannot be applied to realistic layouts which have multiple grid constraints.

This paper discusses extending pitchmatching algorithms to realistic layouts that have multiple grid constraints. Section II is a preliminary section which describes compaction algorithm for layouts with multiple grid constraints. The pitchmatching problem for layouts having multiple grid constraints is formulated in Section III. Section IV presents an algorithm for solving the problem defined in Section III. Also, the computational com-

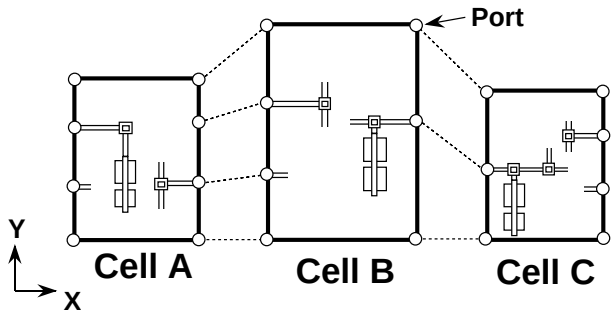

Fig. 1. An example of pitchmatching.

plexity of the proposed algorithm is analyzed in comparison with a naive approach. Section V presents several examples and an application result. Section VI concludes the paper.

II. Compaction with Multiple Grid Constraints

A compaction algorithm that can deal with multiple grid constraints is proposed in [5]. However, that paper only outlines several results without proofs. This section presents the necessary results to formulate pitchmatching problem for layouts with multiple grid constraints. From now on, without loss of generality, we consider compaction in y-direction.

Let $G = (V, E)$ be a constraint graph for y-directional compaction, with a vertex set $V = \{i\}$ and a directed edge set $E = \{(i,j)\} \subset V \times V$. A weight set $W = \{w(i,j) \,|\, (i,j) \in E\}$ is also associated with edge set E of constraint graph G. In the constraint graph $G = (V, E)$, each directed edge (i,j) with weight $w(i,j)$ corresponds to the constraint represented by inequality $y(j) - y(i) \geq w(i,j)$. Here, $y(i)$ denotes the y-coordinate of vertex i. Also let y-coordinate $y(\cdot)$ be an integer because the coordinates of elements in realistic layouts are confined to integer multiples of the most resolvable unit, for example, $0.05\mu m$.

For some vertex $i \in V$, we can define the grid constraint whose form is $< c_{grid}(i), c_{delta}(i) >$ for given positive integer $c_{grid}(i)$ and integer $c_{delta}(i)$, where $c_{grid}(i) \neq 1$ and $|c_{delta}(i)| < c_{grid}(i)$. The grid constraint $< c_{grid}(i), c_{delta}(i) >$ for vertex $i \in V$ implies that y-coordinate $y(i)$ is confined to $c_{grid}(i) \cdot n + c_{delta}(i)$ for some integer n. Let $V_g\,(\subset V)$ be a set of vertices i having the grid constraint denoted by $< c_{grid}(i), c_{delta}(i) >$.

The compaction problem can be formulated by a mixed-integer linear programming problem.

Compaction Problem A

$$
\begin{array}{ll}
\textit{minimize} & y(n) - y(0) \\
\textit{subject to} & 1)\ y(j) - y(i) \geq w(i,j) \\
& \quad \textit{for all edges } (i,j) \in E. \\
& 2)\ y(i) \textit{ satisfies grid constraint} \\
& \quad < c_{grid}(i), c_{delta}(i) > \\
& \quad \textit{for all vertices } i \in V_g.
\end{array}
$$

Here, vertex 0 and n denote the special vertices that correspond to lowermost and uppermost layout elements, respectively. The coordinates of other vertices are bounded by those vertices (i.e., $y(0) \leq y(i) \leq y(n)$ for all $i \in V$). For simplicity, let vertex 0 be the origin of y-coordinate $y(\cdot)$ (i.e., $y(0) = 0$). In the remainder of the paper, **Compaction Problem A** may be abbreviated to **Problem A**.

The solution to **Problem A** is discussed in [5]. The paper presents several algorithms, but includes no proofs that validate them. Thus, in this section, we describe proofs of the claims, which are necessary in the following discussion to make the paper self-contained.

Definition 1 *For integer x and vertex $i \in V$, function $round(x,i)$ with integer value is defined as*

$round(x,i) \stackrel{def}{=} x$ *(if i has no grid constraint (i.e., $i \in V - V_g$)),* $\stackrel{def}{=} n \cdot c_{grid}(i) + c_{delta}(i)$ *(if i has grid constraint $< c_{grid}(i), c_{delta}(i) >$ (i.e., $i \in V_g$) and $(n-1) \cdot c_{grid}(i) + c_{delta}(i) < x \leq n \cdot c_{grid}(i) + c_{delta}(i)$ holds for some integer n.).*

In constraint graph G, consider a path $p = (i_0, i_1, \ldots, i_n)$ that is represented by a sequence of vertices. Let $p_k = (i_0, i_1, \ldots, i_k)$ $(k = 0, 1, \ldots, n)$ be a sub-path from i_0 to i_k. The effective length of the path starting from vertex i_0 with an initial value u_0, which is denoted by $L_e(p_k, u_0, i_0)$ $(k = 0, 1, \ldots, n)$, is defined recursively as

$$
\begin{array}{ll}
L_e(p_0, u_0, i_0) & = u_0, \\
L_e(p_k, u_0, i_0) & = round(L_e(p_{k-1}, u_0, i_0) + w(i_{k-1}, i_k), i_k) \\
& \qquad (k = 1, 2, \ldots, n).
\end{array}
$$

Using the effective length of the path $L_e(\cdot,\cdot,\cdot)$, the longest effective path length from source vertex 0 to vertex i is defined by

$$L_e^{max}(i) = \max\{L_e(p, 0, 0) \,|\, \text{all paths } p \text{ from 0 to } i.\}.$$

Proposition 1 [5] *The minimum solution $\{y(i) \,|\, i \in V\}$ to* **Problem A** *is given by the longest effective path length from source vertex 0 to vertex i (i.e., $y(i) = L_e^{max}(i)$) in G, if there exists a solution to* **Problem A**.

Proof The proof is almost the same as that of Theorem 1 in [4].■

Let c be a cycle with starting vertex i_0 and its initial value u_0. In addition, let c^j denote the j-times union of c, and put $u_j = L_e(c^j, u_0, i_0)$.

The two lemmas below appear in [5] without proofs.

Lemma 1 *There exits a nonnegative integer m and positive integers J, $j1$ such that $u_{j+n\cdot J} = u_j + n \cdot m \cdot D(c)$ for all $j \geq j1$ and $n \geq 0$, where $D(c)$ is the least common multiple of grid constants $\{c_{grid}(i) | \, i \in c \cap V_g\}$.*

Proof The proof is divided into two cases.

(Case 1) The process for calculating the effective length of cycle c recursively ends after executing finite steps.

In this case, since there exists an integer $j1 \geq 1$ such that $u_{j1+n} = u_{j1}$ for all $n \geq 0$, $u_{j+n\cdot J} = u_j + n \cdot m \cdot D(c)$ for all $j \geq j1$ and $n \geq 0$ with $J = 1$ and $m = 0$.

(Case 2) The process for calculating the effective length of cycle c continues infinitely.

In this case, there exists some integers $j < i$ such that $u_i \equiv u_j \pmod{D(c)}$. This implies that $u_i - u_j = m \cdot D(c)$ for some integer $m \geq 1$. Thus, the process for calculating the effective length after u_i is essentially the same as the process from u_j to u_i. Putting $i - j = J(\geq 1)$ and $j = j1$,

$$\begin{aligned} u_{j+n\cdot J} - u_j &= (u_{j+n\cdot J} - u_{j+(n-1)\cdot J}) \\ &\quad +(u_{j+(n-1)\cdot J} - u_{j+(n-2)\cdot J}) \\ &\quad \cdots \\ &\quad +(u_{j+J} - u_j) \\ &= n \cdot m \cdot D(c) \end{aligned}$$

for all $j \geq j1$. Hence, $u_{j+n\cdot J} = u_j + n \cdot m \cdot D(c)$ for all $j \geq j1$ and $n \geq 0$ for some positive integers J, $j1$ and a nonnegative integer m. ∎

On the basis of Lemma 1, effective cycle length is defined as follows.

Definition 2 *The effective cycle length for a cycle is defined as:*

$$L_e(c) = m \cdot D(c)/J.$$

Lemma 2 *The sequence $\{u_j/j\}$ converges to $L_e(c)$ as j tends to infinity.*

Proof Let J and $j1$ be positive integers in Lemma 1. For any sufficiently large $j \geq j1$, there exists an integer n_j such that $j1 + n_j \cdot J \leq j < j1 + (n_j + 1) \cdot J$, and k_j such that $0 \leq k_j \leq J - 1$ and $j = j1 + n_j \cdot J + k_j$. Without loss of generality, let $u_j \geq 0$. Since $n_j \to \infty$ as $j \to \infty$ and k_j is bounded,

$$\begin{aligned} (u_j/j) &= (u_{j1+k_j} + n_j \cdot m \cdot D(c))/(j1 + n_j \cdot J + k_j) \\ &= ((u_{j1+k_j}/n_j) + m \cdot D(c))/(((j1 + k_j)/n_j) + J) \\ &\to m \cdot D(c)/J \quad (n_j \to \infty) \end{aligned}$$

Hence, $u_j/j \to m \cdot D(c)/J \quad (j \to \infty)$ holds. ∎

Definition 3 *A cycle c is an effectively positive cycle if and only if $L_e(c) > 0$, or equivalently $\lim_{j\to\infty} u_j/j > 0$.*

Proposition 2 *The necessary and sufficient condition for the existence of a sulution to* **Problem A** *is that there is no effectively positive cycle in the constraint graph G.*

Proof From Proposition 1, Definition 2 and the proof of Lemma 1 (Case 1), this proposition can be proved. ∎

The following definition and lemma are corrections of [5].

Definition 4 *For each vertex $i \in V$, bounding number b_i is defined by*

$$b_i \stackrel{def}{=} \begin{cases} D/c_{grid}(i) & (\textit{if } i \in V_g) \\ \sum_{j \in V_g} D/c_{grid}(j) + 1 & (\textit{if } i \in V - V_g), \end{cases}$$

where D is the least common multiple of grid constants $\{c_{grid}(i) \, | \, i \in V_g\}$.

Lemma 3 [5] *If there is no effctively positive cycle in G, then the effectively longest path lengths can be found among paths which do not traverse any vertex $i \in V$ more than b_i times.*

Proof Two assertions are prooved.

(a) Any effectively longest path cannot include grid vertex $i \in V_g$ more than b_i times.

Assume that some effectively longest path traverses grid vertex i at $(b_i + 1)$ times. Let c_j be a path which is traversed from the $(j-1)$-th to the j-th traverse of vertex i in the effectively longest path. Here, let the 0-th traverse of vertex i imply the start from source vertex 0. In addition, putting $c^j = c_1 \cup c_2 \cup \ldots \cup c_j$, let $u_j = L_e(c^j, 0, 0)$. If $u_i = u_j$ for i, j such that $1 \leq i, j \leq b_i + 1$, the number of traverses of vertex i can be decreased by 1 when an unnecessary cycle is deleted. Thus, we assume $u_1 < u_2 < \ldots < u_{b_i+1}$. Since vertex i has a grid constraint $< c_{grid}(i), c_{delta}(i) >$, $u_j = c_{grid}(i) \cdot n_j + c_{delta}(i) \quad (j = 1, 2, \ldots, b_i + 1)$ and $n_{j_k} \neq n_{j_l}$ for j_k, j_l such that $j_k \neq j_l$. Thus, for some j_1 and j_2 such that $j_1 < j_2$, $n_{j_1} \equiv n_{j_2} \pmod{b_i}$. This implies $u_{j_1} \equiv u_{j_2} \pmod{D}$, or equivalently $u_{j_2} - u_{j_1} = m \cdot D$ for some integer $m \geq 1$. This leads to the relation $u_{j+n\cdot(j_2-j_1)} = u_j + n \cdot m \cdot D$ for all $j \geq j_1$ and $n \geq 0$. This implies the existence of an effectively positive cycle, which is a contradiction. This proves assertion (a).

(b) Any effectively longest path cannot include non-grid vertex $i \in V - V_g$ more than b_i times.

It should be mentioned that an effectively longest path does not traverse a non-grid vertex i twice without passing a certain grid vertex between the two traverses. This is because if an effectively longest path traverses a non-grid vertex i twice without intervening traverses of grid vertices, there exists an effectively positive cycle. For non-grid vertex i, the effectively longest path which traverses vertex i most repeatedly has the form as shown in Fig. 2 because this path cannot traverse grid vertex j more than $b_j (= D/c_{grid}(j))$ times from the proof of assertion (a). Hence, the number of traverses of vertex i is summed up to $b_i = \sum_{j \in V_g} D/c_{grid}(j) + 1$. ∎

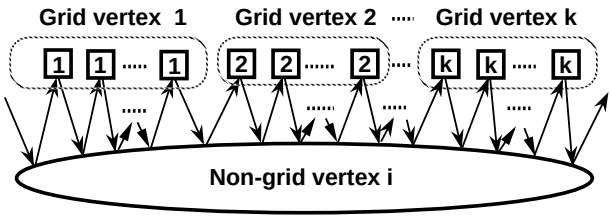

Fig. 2. An effectively longest path that traverses non-grid vertex i most repeatedly.

III. The Pitchmatching Problem

Many realistic layout problems often require additional constraints besides constraints 1) and 2) in **Problem A**. For example, $y(i) - y(j) = y(k) - y(l)$ for vertices $i, j, k, l \in V$. This constraint, known as symmetry constraint, makes two specified distances between layout elements equal (See [9]). This kind of constraint is often applied to analog circuit layouts. On the other hand, constraint $y(i) = y(j)$ for some pairs of vertices $i, j \in V$ is used for pitchmatching [1]. This type of constraint enables designers to construct macro cells from constituent leaf cells because the pitchmatching can make y-coordinates of leaf cell ports equal making interconnect of those ports possible as illustrated in Fig. 1.

This section formulates the pitchmatching problem for layouts with multiple grid constraints. However, the compaction problem with symmetry constraints can be dealt with in similar fashion.

From here on, we consider **Compaction Problem** $\tilde{A}$, to which is added a third constraint 3).

Compaction Problem $\tilde{A}$

minimize $y(n) - y(0)$
subject to
1) $y(j) - y(i) \geq w(i,j)$
for all edges $(i,j) \in E$.
2) $y(i)$ *satisfies grid constraint*
$< c_{grid}(i), c_{delta}(i) >$
for all vertices $i \in V_g$.
3) $y(i) = y(j)$
for vertex-pairs $(i,j) \in P \subset V \times V$.

Here, a set of vertex-pairs P must be specified in advance.

In what follows, let V_c be a set of vertices defined as

$$V_c = \{\, i \in V \mid (i,j) \in P \text{ or } (j,i) \in P \text{ for some } j \in V \,\} \cup \{0\} \cup \{n\}.$$

This definition implies that V_c is the set of vertices relevant to vertex-pairs $P \subset V \times V$.

procedure $L_{s,t}(\delta)$;

```
(1) begin
(2)    for (all vertices i ∈ V) do y(i) ← −∞;
(3)    y(s) ← round(δ, s);
(4)    for (ind ← 1) to N do
(5)       for (all edges (i, j) ∈ E) do
(6)       if (y(j) < y(i) + w(i, j)) then
(7)         begin
(8)            y(j) ← y(i) + w(i, j);
(9)            if (j ∈ V_g) then y(j) ← round(y(j), j);
(10)        end
(11)   for (all edges (i, j) ∈ E) do
(12)      if (y(j) < y(i) + w(i, j))then
(13)      begin
(14)         print ("constraint conflict detected.");
(15)         stop;
(16)      end
(17)   return y(t);
(18)    /* {y(i) | i ∈ V} is obtained. */
(19) end
```

Fig. 3. Flow of **Procedure** $L_{s,t}(\delta)$.

Definition 5 *For every vertex-pair* (i,j), *distance* $d(i,j)$ *is defined as*

$$d(i,j) = \begin{cases} 0 & (\text{ if } i = j) \\ -\infty & (\text{ if no path from } i \text{ to } j \text{ exists.}) \\ \min_{0 \leq \delta \leq D}(L_{i,j}(\delta) - \delta) & \\ \qquad (\text{ if path from } i \text{ to } j \text{ exists.}), & \end{cases}$$

where $L_{i,j}(\delta)$ *is calculated by* **procedure** $L_{s,t}(\delta)$ *in Fig. 3 with* $s = i$ *and* $t = j$. *Positive integer* D *is the least common multiple of grid constants* $\{c_{grid}(i) \mid i \in V_g\}$.

Proposition 3 *If constraint graph* G *includes no effectively positive cycles, then procedure* $L_{i,j}(\delta)$ *returns a definite value when the limit of loops* N *is defined by* $N = \sum_{i \in V_g} b_i + (\sum_{i \in V_g} b_i + 1) \cdot |V - V_g| - 1$.

From **Compaction Problem** $\tilde{A}$ we make a new compaction problem associated with a set of core vertices V_c using distance $d(\cdot,\cdot)$ defined in Definition 5.

Core Compaction Problem $\tilde{A}_c$

minimize $y(n) - y(0)$
subject to
1) $y(j) - y(i) \geq d(i,j)$
for all vertices $i, j \in V_c$.
2) $y(i)$ *satisfies grid constraint*
$< c_{grid}(i), c_{delta}(i) >$
for all vertices $i \in V_g \cap V_c$.
3) $y(i) = y(j)$
for all vertex-pairs $(i,j) \in P \subset V \times V$.

Equivalent Compaction Problem $\tilde{A}_e$

Let $\{\tilde{y}_c(i) \mid i \in V_c\}$ be a solution to **Core Compaction Problem** $\tilde{A}_c$.

minimize $y(n) - y(0)$
subject to 1) $y(j) - y(i) \geq w(i,j)$ *for all edges* $(i,j) \in E$.
2) $y(i)$ *satisfies grid constraint* $< c_{grid}(i), c_{delta}(i) >$ *for all vertices* $i \in V_g$.
3) $y(i) = \tilde{y}_c(i)$ *for every vertex* $i \in V_c$.

Lemma 4 *For arbitrary* $\delta \geq 0$ *and path* $p = (i_1, i_2, i_3, \ldots, i_{p-1}, i_p)$ *of* G,

$$L_{i_1,i_p}(\delta) \geq L_{i_{p-1},i_p}(L_{i_{p-2},i_{p-1}}(\cdots(L_{i_2,i_3}(L_{i_1,i_2}(\delta)))\cdots))$$

holds.

Proof Clearly,

$$L_{i_1,i_p}(\delta) = \max_p\{L_{j_k,i_p}(L_{j_{k-1},j_k}(\cdots(L_{j_1,j_2}(L_{i_1,j_1}(\delta)))\cdots))\},$$

where the maximum is taken over all paths $p = (i_1, j_1, j_2, \ldots, j_k, i_p)$ from i_1 to i_p and $k \geq 1$. Thus, if we select a path $p = (i_1, i_2, i_3, \ldots, i_{p-1}, i_p)$, then

$$L_{i_1,i_p}(\delta) \geq L_{i_{p-1},i_p}(L_{i_{p-2},i_{p-1}}(\cdots(L_{i_2,i_3}(L_{i_1,i_2}(\delta)))\cdots))$$

holds. ■

Proposition 4 *For arbitrary vertices* $i, j, k \in V$, *inequality* $d(i,k) \geq d(i,j) + d(j,k)$ *holds.*

Proof If $d(i,k) = -\infty$, then there is no path from vertex i to k. Thus, either there is no path from i to j or there is no path from j to k. This implies $d(i,j) = -\infty$ or $d(j,k) = -\infty$. Since the right-hand side of the inequality becomes $-\infty$, the inequality holds. Thus, we assume that $d(i,k) > -\infty$. If $d(i,j) = -\infty$ or $d(j,k) = -\infty$, then the inequality holds. Thus, we also assume $d(i,j) > -\infty$ and $d(j,k) > -\infty$.

Let δ_0 be an integer such that $L_{i,k}(\delta_0) - \delta_0 = \min_{0 \leq \delta \leq D}(L_{i,k}(\delta) - \delta) \stackrel{def}{=} d(i,k)$ holds. Since $L_{i,j}(\delta_0) - \delta_0 \geq d(i,j)$ from Definition 5 and $L_{i,k}(\delta_0) \geq L_{j,k}(L_{i,j}(\delta_0))$ from Lemma 4, for this δ_0,

$$\begin{aligned} d(i,k) &= L_{i,k}(\delta_0) - \delta_0 \\ &= (L_{i,j}(\delta_0) - \delta_0) + (L_{i,k}(\delta_0) - L_{i,j}(\delta_0)) \\ &\geq d(i,j) + (L_{j,k}(L_{i,j}(\delta_0)) - L_{i,j}(\delta_0)). \end{aligned}$$

Furthermore, we obtain $L_{j,k}(L_{i,j}(\delta_0)) - L_{i,j}(\delta_0) \geq d(j,k)$ from the definition of $d(\cdot,\cdot)$. Hence, $d(i,k) \geq d(i,j) + d(j,k)$ holds. ■

As in [10], we consider two conditions below.

Condition 1 For an arbitrary feasible solution $\tilde{s}_c = \{\tilde{y}_c(i) \mid i \in V_c\}$ to **Core Problem** $\tilde{A}_c$, there exists a feasible solution $\tilde{s} = \{\tilde{y}(i) \mid i \in V\}$ to **Problem** $\tilde{A}$ such that 1) $\tilde{y}(i) = \tilde{y}_c(i)$ for all vertices $i \in V_c$, (i.e., the restriction of $\tilde{s}$ to V_c coincides with $\tilde{s}_c$, which is denoted by $\tilde{s} \mid_{V_c} = \tilde{s}_c$), and 2) the values of objective functions (i.e., $y(n) - y(0)$) for $\tilde{s}$ and $\tilde{s}_c$ coincide.

Condition 2 For an arbitrary feasible solution $\tilde{s} = \{\tilde{y}(i) \mid i \in V\}$ to **Problem** $\tilde{A}$, $\tilde{s}_c = \{\tilde{y}(i) \mid i \in V_c\}$ (i.e., $\tilde{s}_c = \tilde{s} \mid_{V_c}$) is a feasible solution to **Core Problem** $\tilde{A}_c$ such that the values of objective functions (i.e., $y(n) - y(0)$) for $\tilde{s}$ and $\tilde{s}_c$ coincide.

The two propositions below appear in [10].

Proposition 5 *Assume that* **Conditions 1** *and* **2** *hold. If there exists an optimum solution* $\tilde{s}_c^{opt} = \{\tilde{y}_c^{opt}(i) \mid i \in V_c\}$ *to* **Core Problem** $\tilde{A}_c$, *then there exits an optimum solution* $\tilde{s}^{opt} = \{\tilde{y}^{opt}(i) \mid i \in V\}$ *to* **Problem** $\tilde{A}$ *such that 1)* $\tilde{s}^{opt} \mid_{V_c} = \tilde{s}_c^{opt}$, *and 2) the values of the objective function for* $\tilde{s}_c^{opt}$ *and* $\tilde{s}^{opt}$ *coincide.*

Proof Let $\tilde{s}_c^{opt} = \{\tilde{y}_c^{opt}(i) \mid i \in V_c\}$ be an optimal solution to **Problem** $\tilde{A}_c$, which has optimum value $\tilde{v}^{opt}$. From **Condition 1**, there exists a feasible solution $\tilde{s} = \{\tilde{y}(i) \mid i \in V\}$ to **Problem** $\tilde{A}$ such that 1) $\tilde{s} \mid_{V_c} = \tilde{s}_c$ and 2) the optimum value is $\tilde{v}^{opt}$. If there exists a feasible solution $\tilde{s}'$ to **Problem** $\tilde{A}$ whose objective function value is $\tilde{v}' (< \tilde{v}^{opt})$, then from **Condition 2**, $\tilde{s}' \mid_{V_c}$ is a feasible solution to **Problem** $\tilde{A}_c$ having the objective function value is $\tilde{v}' (< \tilde{v}^{opt})$. This contradicts the optimality of $\tilde{s}_c^{opt}$ to **Problem** $\tilde{A}_c$. This implies that $\tilde{s}$ is the optimum solution to **Problem** $\tilde{A}$. ■

Proposition 6 *If* **Conditions 1** *and* **2** *hold, then the optimum solution to* **Equivalent Problem** $\tilde{A}_e$ *is the optimum solution to* **Problem** $\tilde{A}$.

Proof Assume that **Conditions 1** and **2** hold. Let $\tilde{s}_e$ be an optimum solution to **Equivalent Problem** $\tilde{A}_e$ whose objective function value is v. This implies the existence of the optimum solution $\tilde{s}_c^{opt}$ to **Core Problem** $\tilde{A}_c$ which has also optimum value v. From Proposition 5, there exists an optimum solution $\tilde{s}^{opt}$ to **Problem** $\tilde{A}$ such that $\tilde{s}^{opt} \mid_{V_c} = \tilde{s}_c^{opt}$ and the optimum value is v.

Clearly, $\tilde{s}_e$ is also a feasible solution to $\tilde{A}$. The objective function value for $\tilde{s}_e$ is v which is the optimum value of **Problem** $\tilde{A}$. Hence, $\tilde{s}_e$ is the optimum solution to $\tilde{A}$. ■

Definition 6 *For an arbitrary feasible solution* $\tilde{s}_c = \{\tilde{y}_c(i) \mid i \in V_c\}$ *to* **Core Problem** $\tilde{A}_c$, *the extension of* $\tilde{s}_c$ *to* V *is defined as* $\tilde{s} = \{\tilde{y}(j) \mid j \in V\}$ *such that*

$$\tilde{y}(j) \stackrel{def}{=} \max_{i \in V_c} round(\tilde{y}_c(i) + d(i,j), j)$$

for all vertices $j \in V$. *For simplicity, we call this the extension of* $\tilde{s}_c$.

Lemma 5 *Let* $\tilde{s} = \{\tilde{y}(i) \mid i \in V\}$ *be an extension of* $\tilde{s}_c = \{\tilde{y}_c(i) \mid i \in V_c\}$. *Assertions 1), 2), and 3) below are satisfied:*

1) The relation $\tilde{y}(i) = \tilde{y}_c(i)$ holds for all vertices $i \in V_c$ (i.e., $\tilde{s} \mid_{V_c} = \tilde{s}_c$).

2) The relation $\tilde{y}(j) - \tilde{y}(i) \geq w(i,j)$ holds for every edge $(i,j) \in E$ such that $i \in V - V_g$.

3) If $\tilde{y}(j) - \tilde{y}(i) \geq w(i,j)$ holds for every edge $(i,j) \in E$ such that $i \in V_g$, then **Condition 1** *is satisfied.*

Proof of 1) Since $\tilde{s}_c$ is a feasible solution to $\tilde{A}_c$, $\tilde{y}_c(j) - \tilde{y}_c(i) \geq d(i,j)$ for all vertices $i,j \in V_c$. Thus, $\tilde{y}_c(j) \geq round(\tilde{y}_c(i) + d(i,j), j)$ for all vertices $i,j \in V_c$. This leads to $\tilde{y}_c(j) \geq \max_{i \in V_c} round(\tilde{y}_c(i) + d(i,j), j)$ for all vertices $j \in V_c$. On the other hand, $\max_{i \in V_c} round(\tilde{y}_c(i) + d(i,j), j) \geq round(\tilde{y}_c(j) + d(j,j), j) \geq \tilde{y}_c(j)$ for all vertices $j \in V_c$. This implies that $\tilde{y}(j) = \tilde{y}_c(j)$ for all vertices $j \in V_c$.

Proof of 2) Let $(i,j) \in E$ be an edge such that $i \in V - V_g$. From Definition 6, $\tilde{y}(i) = \max_{k \in V_c} (\tilde{y}_c(k) + d(k,i))$. Let $k' \in V_c$ be a vertex such that $\tilde{y}(i) = \tilde{y}_c(k') + d(k',i)$. For this $k' \in V_c$,

$$\begin{aligned}
\tilde{y}(j) - \tilde{y}(i) &= \max_{l \in V_c} round(\tilde{y}_c(l) + d(l,j), j) \\
&\quad -(\tilde{y}_c(k') + d(k',i)) \\
&\geq round(\tilde{y}_c(k') + d(k',j), j) \\
&\quad -(\tilde{y}_c(k') + d(k',i)) \\
&\geq (\tilde{y}_c(k') + d(k',j)) - (\tilde{y}_c(k') + d(k',i)) \\
&= d(k',j) - d(k',i) \\
&\geq d(i,j) \\
&\geq w(i,j).
\end{aligned}$$

This completes the proof.

Proof of 3) From the validity of assertions 1) and 2), it is clear that assertion 3) holds. ∎

Proposition 7 **Condition 2** *is always satisfied.*

Proposition 8 *If $V_g \subset V_c$, then* **Condition 1** *holds.*

Proof Let $(i,j) \in E$ be an edge such that $i \in V_g$ and $\tilde{s} = \{\tilde{y}(i) \mid i \in V\}$ be an extension of $\tilde{s}_c = \{\tilde{y}_c(i) \mid i \in V_c\}$. Since $V_g \subset V_c$, then $i \in V_c$. Thus, from assertion 1) of Lemma 5, $\tilde{y}(i) = \tilde{y}_c(i)$. This leads to the relation

$$\begin{aligned}
\tilde{y}(j) - \tilde{y}(i) &= \max_{l \in V_c} round(\tilde{y}_c(l) + d(l,j), j) - \tilde{y}_c(i) \\
&\geq round(\tilde{y}_c(i) + d(i,j), j) - \tilde{y}_c(i) \\
&\geq (\tilde{y}_c(i) + d(i,j)) - \tilde{y}_c(i) \\
&= d(i,j) \\
&\geq w(i,j).
\end{aligned}$$

Consequently, from assertion 3) of Lemma 5, **Condition 1** holds. ∎

IV. Pitchmatching Algorithms

A. Pitchmatching Model

On the basis of the formulation in Section III, we consider pitchmatching algorithms that can be applied to realistic layouts having multiple grid constraints. Fig. 4 shows the layout model to which the pitchmatching algorithms are applied. The entire macro cell is constructed by placing adjacent leaf cells C^i $(i = 1,2,\ldots,n)$ in the x-direction. Each leaf cell layout must be shrunk or spread in the y-direction so that corresponding ports of adjacent leaf cells can be placed at the same y-coordinates. In Fig. 4, broken lines denote port-pairs which must be pitchmatched.

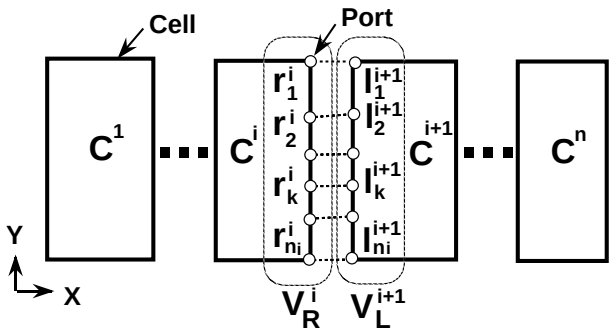

Fig. 4. Pitchmatching model. Ports corresponding to vertices $r_k^i \in V_R^i$ and $l_k^{i+1} \in V_L^{i+1}$ must have the same y-coordinates for all $k (1 \leq k \leq n_i)$.

Let $G^i = (V^i, E^i)$ be a constraint graph for cell C^i. A set of vertices having grid constraints in V^i is denoted by V_g^i. Further, let V_L^i (V_R^i) be a set of vertices placed on the left (right)-hand side boundary of C^i. Assume that the number of vertices in V_R^i is the same as that in V_L^{i+1} $(i = 1,2,\ldots,n-1)$, say n_i. Thus we can represent V_R^i (V_L^{i+1}) $(i = 1,2,\ldots,n-1)$ by $V_R^i = \{r_1^i, r_2^i, \ldots, r_{n_i}^i\}$ $(V_L^{i+1} = \{l_1^{i+1}, l_2^{i+1}, \ldots, l_{n_i}^{i+1}\})$. All pairs of ports corresponding to those of vertices (r_k^i, l_k^{i+1}) $(k = 1,2,\ldots,n_i)$ in V_R^i and V_L^{i+1} must be pitchmatched.

The constraint graph $G = (V,E)$ for the entire cell can be constructed by connecting constraint graphs $G^i (i = 1,2,\ldots,n)$ with edge sets $E(i) (i = 1,2,\ldots,n-1)$. For simplicity, we put $E(n) = \emptyset$. Here, $E(i)$ is the set of directed edges between leaf cells C^i and C^{i+1}, which includes pairs of edges (r_k^i, l_k^{i+1}) and (l_k^{i+1}, r_k^i) $(r_1^i, r_2^i, \ldots, r_{n_i} \in V_R^i;\ l_1^{i+1}, l_2^{i+1}, \ldots, l_{n_i}^{i+1} \in V_L^{i+1})$ with reverse directions and weights of zero. The constraints caused by these edge-pairs add constraints $y(r_k^i) = y(l_k^{i+1})$ $(k = 1,2,\ldots,n_i; i = 1,2,\ldots,n-1)$. Hence, solving constraint graph G corresponds to **Compaction Problem** $\tilde{A}$ with $P = \cup_{i=1}^n E(i)$ in Section III.

B. An Extension to Multiple Grid Constraints

We describe an algorithm for solving constraint graph G defined above. In what follows, we refer to it as **EXTENT**.

Initially, we generate the **Core Compaction Prob-**

lem $\tilde{A}_c$ from constraint graphs G^i $(i = 1, 2, \ldots, n)$. Let the constraint graph for **Problem** $\tilde{A}_c$ be $G_c = (V_c, E_c)$. From Proposition 8, vertex set V_c must include all grid vertices V_g $(= \cup_{i=1}^{n} V_g^i)$. Thus, V_c is defined as $V_c = V_g \cup (\cup_{i=1}^{n} V_L^i) \cup (\cup_{i=1}^{n} V_R^i)$. The distance $dis(\cdot, \cdot)$ defined in Definition 5 is calculated in constraint graph G. As indicated in [9][10], if there are no grid vertices, $d(\cdot, \cdot)$ can be calculated only within each G^i. On the other hand, if there are grid vertices, $d(\cdot, \cdot)$ must be calculated over entire G, as detailed in example 2 of Section V. However, in order to reduce processing times, we use a similar method to that applied to the problem having no grid vertices. The procedure is described in Fig. 5.

As shown in Fig. 5, for each $i\,(i = 1, 2, \ldots, n)$, we generate constraint graph G_c^i which corresponds to the **Core Compaction Problem** of G^i (See from (4) to (20) in Fig. 5.). After that, those constraint graphs $G_c^i\,(i = 1, 2, \ldots, n)$ are merged into a single constraint graph $G_c = (V_c, E_c)$ having additional edge sets $E(i)\,(i = 1, 2, \ldots, n-1)$. The **Core Problem** $\tilde{A}_c$ is defined by graph G_c.

Next, the **Core Problem** $\tilde{A}_c$ is solved by the same algorithm as **procedure** $L_{s,t}(\delta)$ with $\delta = 0$. It should be noted that **procedure** $L_{s,t}(\delta)$ can obtain $y(i)$ for vertices $i \in V_c$. This yields the solution $\tilde{s}_c = \{\tilde{y}_c(i) \,|\, i \in V_c\}$ to **Problem** $\tilde{A}_c$

Finally, in order to fix the coordinates of vertices in V_c, we add constraint $\{y(i) = \tilde{y}_c(i) \,|\, i \in V_c\}$ into constraint graph G. This constraint graph corresponds to the **Equivalent Compaction Problem** $\tilde{A}_e$. Again, we apply the same algorithm as **procedure** $L_{s,t}(\delta)$ with $\delta = 0$ to G. Consequently, we obtain the solution $\tilde{s} = \{\tilde{y}(i) \,|\, i \in V\}$ to the original problem $\tilde{A}$ whose constraint graph is G.

Having described the processing flow of **EXTENT**, let us now turn to the analysis of its time complexity. Clearly, the time complexities for solving the **Core Problem** $\tilde{A}_c$ and the **Equivalent Problem** $\tilde{A}_e$ are equal to the time complexity of **procedure** $L_{s,t}(\delta)$. However, in the phase for generating **Core Problem** $\tilde{A}_c$, the **procedure** $L_{s,t}(\delta)$ is performed $|V_c^i|$ times for all the constraint graphs G^i $(i = 1, 2, \ldots, n)$. Since **procedure** $L_{s,t}(\delta)$ for each constraint graph G^i has the time complexity $O(D \cdot |E^i| \cdot N_i)$, the total time complexity becomes $T_{EXTENT} = O(\sum_{i=1}^{n} |V_c^i| \cdot D \cdot |E^i| \cdot N_i)$. Here, N_i is the number defined as N in Proposition 3 for constraint graph G^i.

C. A Naive Approach

The simplest approach to the pitchmatching problem is to apply **procedure relaxation** in Fig. 6 to constraint graph G.

From now on, we refer to this algorithm as **RELAX**. This procedure is essentially equivalent to procedure $L_{s,t}(\delta)$ in Fig. 3, if δ is set at zero and s is a ver-

procedure core;

```
(1) begin
(2) /* G_c^i = (V_c^i, E_c^i): constraint graph */
(3) /* for core compaction problem of C^i */
(4)    for (i ← 1) to n do
(5)    begin
(6)       V_c^i ← V_L^i ∪ V_R^i ∪ V_g^i;
(7)       E_c^i ← ∅;
(8)       for (all vertices s ∈ V_c^i) do
(9)       for (δ ← 0) to D do
(10)         call procedure L_{s,.}(δ) with respect to
(11)         constraint graph G^i = (V^i, E^i);
(12) /* {L_{s,t}(δ)| s,t ∈ V_c^i; 0 ≤ δ ≤ D} is obtained. */
(13)      for (all vertices s ∈ V_c^i) do
(14)      for (all vertices t(≠ s) ∈ V_c^i) do
(15)      begin
(16)         if (there is a path from s to t) then
(17)         w_{s,t} ← min_{0≤δ≤D}(L_{s,t}(δ) − δ);
(18)         E_c^i ← E_c^i ∪ {(s,t) with weight w_{s,t}};
(19)      end
(20)   end
(21) /*  G_c = (V_c, E_c): constraint graph for   */
(22) /*  core compaction problem of entire cell C.   */
(23)   construct constraint graph G_c
(24)   by merging G_c^i(i = 1, 2, ..., n) with
(25)   edge sets E(i) (i = 1, 2, ..., n − 1);
(26) end
```

Fig. 5. Flow of **Procedure core**. This procedure generates **Core Problem**

procedure relaxation;

```
(1) begin
(2)    for (all vertices i ∈ V) do y(i) ← −∞;
(3)    for (all vertices s ∈ V corresponding to
(4)    the lowermost elements in C^i (i = 1, 2, ..., n))
(5)    do y(s) ← 0;
(6)    for (ind ← 1) to N do
(7)    for (i ← 1) to n do
(8) /*  Relaxation for cell C^i which has  */
(9) /*  constraint graph G^i = (V^i, E^i). */
(10)   begin
(11) /*  E(i): edges between C^i and C^{i+1}.  */
(12)      for (all edges (s,t) ∈ E^i ∪ E(i)) do
(13)      if (y(t) < y(s) + w(s,t)) then
(14)      begin
(15)         y(t) ← y(s) + w(s,t);
(16)         if (t ∈ V_g) then y(t) ← round(y(t), t);
(17)      end
(18)   end
(19) end
```

Fig. 6. Flow of **Procedure relaxation**. This is applied to a naive approach to pitchmatching problem.

tex corresponding to the lowermost elements in cell C. Steps (5) to (10) in Fig. 3 are performed in steps (7) to (18) in Fig. 6. In the former procedure, all edges in G are examined in a single **for** loop, while in the latter procedure, all edges are examined through two-folded **for** loops with respect to all cells and the edges in each cell. Since the latter needs to process only constraint graph G^i at any stage, area requirements for processing can be reduced. However, the time complexities for both procedures are the same, which can be denoted by $T_{RELAX} = O((\sum_{i=1}^{n} |E^i|) \cdot N) = O(|E| \cdot N)$. Here, N is defined as in Proposition 3 for constraint graph G.

D. Comparison of Time Complexity

We compare the time complexity of the extended algorithm **EXTENT** with that of the naive algorithm **RELAX**. For simplicity, we assume that constraint graph G consists of n graphs G^i $(i = 1, 2, \ldots, n)$ which have approximately the same sizes. This implies that $|E^i| \simeq |E|/n$. Furthermore, let α $(0 < \alpha < 1)$ be the ratio of the number of vertices having grid constraints to the number of total vertices of the constraint graph. Since $V_c^i = V_L^i \cup V_R^i \cup V_g^i$ and $|V_g^i| \gg |V_L^i|$, $|V_R^i|$ in realistic layouts, we assume that $|V_c^i| \simeq \alpha \cdot |V|/n$. Also, let N_i be approximately proportional to the number of vertices in the relevant constraint graph. This implies $N_i \simeq N/n$. These assumptions lead to the estimation

$$\begin{aligned} T_{EXTENT} &= O(\sum_{i=1}^{n} |V_c^i| \cdot D \cdot |E^i| \cdot N_i) \\ &\simeq O(\alpha \cdot D \cdot (|V|/n^2) \cdot |E| \cdot N) \\ &\simeq (\alpha \cdot D \cdot |V|/n^2) \cdot T_{RELAX}. \end{aligned}$$

Consequently, algorithm **EXTENT** is superior to algorithm **RELAX** only when α is sufficiently small.

V. Examples and Discussion

This section presents several examples that assure the results in Sections III and IV. It also discusses results of applying the pitchmatching algorithm to a realistic layout.

A. Example 1

Fig. 7 shows an example that illustrates the implication of Proposition 8. Fig. 7 (a) is a constraint graph G for a cell C^i. The numbers j in squares denote vertices having grid constraints $< c_{grid}(j), c_{delta}(j) >$ (i.e., $j \in V_g$), while the numbers k in circles correspond to vertices having no grid constraints (i.e., $k \in V - V_g$). Let V_c be a set of vertices {1, 2, 7 ,8} corresponding to the ports on the left- and right-hand sides of cells. Fig. 7 (b) denotes the constraint graph $G_c = (V_c, E_c)$ obtained from original constraint graph G. This graph corresponds to

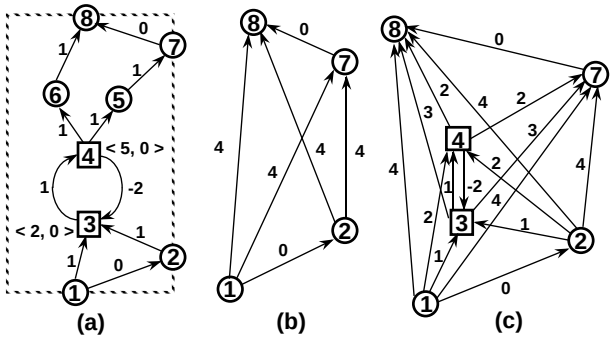

Fig. 7. An example of constraint graph that corresponds to the **Core Problem**. Grid vertices must be included in the constraint graph. (a) Original constraint graph. (b) Constraint graph for only ports. (c) Constraint graph to which grid vertices are added.

TABLE I
Calculated $L_{1,7}(\delta)$ $(0 \le \delta \le D)$ in Fig. 7

δ	0	1	2	3	4	5	6	7	8
$L_{1,7}(\delta)$	7	7	7	7	12	12	12	12	17
$L_{1,7}(\delta) - \delta$	7	6	5	4	8	7	6	5	9

δ	9	10
$L_{1,7}(\delta)$	17	17
$L_{1,7}(\delta) - \delta$	8	7

the **Core Problem**. The weights of edges (i.e., $d(\cdot,\cdot)$) are calculated from Definition 5. For example, the $d(1,7)$ is calculated from the values of $L_{1,7}(\delta) - \delta$ $(0 \le \delta \le D)$ shown in Table I, where $D = 10$. This calculation leads to $d(1,7) = \min_{0 \le \delta \le D}(L_{1,7}(\delta) - \delta) = 4$.

However, the constraint graph in Fig. 7 (b) loses the effective longest path lengths in the original graph G. For example, the effective longest path length from vertex 1 to 7 is seven in G, whereas effective longest path length from 1 to 7 in Fig. 7 (b), it is four. Thus, in order to obtain the **Core Problem** which is equivalent to G, we add all vertices in V_g to V_c as assured in Proposition 8. This yields the constraint graph in Fig. 7 (c). For example, the weight of the edge from vertex 3 to 7 is calculated from the values of $L_{3,7}(\delta) - \delta$ $(0 \le \delta \le D)$ shown in Table II, which is $d(3,7) = \min_{0 \le \delta \le D}(L_{3,7}(\delta) - \delta) = 3$.

B. Example 2

Fig. 8 shows the part of the constraint graph G between cells C^i and C^{i+1}. The broken line denotes the boundary of cells C^i and C^{i+1}. Vertex 1 (3) belongs to cell C^i (C^{i+1}). As in Figs. 2 and 7, squares and circles represent vertices with grid constraints and no grid constraints, respectively. Vertex 2 is a port to be pitch-

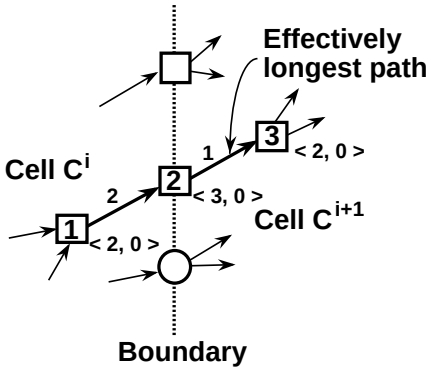

Fig. 8. An example of effectively longest path from vertex 1 to 3 on which relation $d(1,3) \neq d(1,2) + d(2,3)$ holds.

matched between C^i and C^{i+1}. For simplicity, vertex 2 is constructed by merging two ports of C^i and C^{i+1} which must be pitchmatched (i.e., $r_k^i \in V_R^i$ and $l_k^{i+1} \in V_L^{i+1}$ for some k $(1 \leq k \leq n_i)$).

Assume that vertices 1, 2 and 3 belong to V_c. When we generate a constraint graph corresponding to the **Core Problem**, we must calculate the distances $d(\cdot,\cdot)$ between vertices in V_c. If the distances must be calculated for all pairs of vertices in V_c, a large amount of processing time is necessary. Thus, unless the distance calculations can be restricted to the cell areas including the relevant vertex pairs, pitchmatching algorithms cannot be applied to realistic layouts.

Let the distance from vertex 1 to 3 through the path passing bold arrows be $d(1,3)$, which is obtained from Definition 5. It should be remarked that this path includes vertex 2 in V_c. If distance $d(1,3) = d(1,2) + d(2,3)$ holds, the constraint $y(3) \geq y(1) + d(1,3)$ can be replaced with constraints $y(2) \geq y(1) + d(1,2)$ and $y(3) \geq y(2) + d(2,3)$ which can be calculated only within cell C^i and C^{i+1}, respectively.

From Proposition 4 in Section 3, the inequality $d(1,3) \geq d(1,2) + d(2,3)$ always holds. If there are no grid constraints, $d(1,3) = d(1,2) + d(2,3)$ holds because $d(\cdot,\cdot)$ coincides with the usual longest path length. On the other hand, if there are grid constraints, $d(1,3) \neq d(1,2) + d(2,3)$. For example, in Fig. 8, $d(1,3) = 4$, $d(1,2) = 2$, $d(2,3) = 1$. Since we impose those additional constraints on realistic layouts that separate layout elements at certain distances from cell boundaries, the limitation of the distance calculations to each cell causes no design rule violations.

C. Application results

We developed a compaction-based environment for layout design, where we can convert previously designed layouts to those which obey a specified set of design rules.

TABLE II
CALCULATED $L_{3,7}(\delta)$ $(0 \leq \delta \leq D)$ IN FIG. 7

δ	0	1	2	3	4	5	6	7	8
$L_{3,7}(\delta)$	7	7	7	7	7	12	12	12	12
$L_{3,7}(\delta) - \delta$	7	6	5	4	3	7	6	5	4

δ	9	10
$L_{3,7}(\delta)$	17	17
$L_{3,7}(\delta) - \delta$	8	7

The pitchmatching program, in which the algorithms described in this paper are implemented, is an integral part of the compaction programs. The pitchmatching program was applied to various kinds of cells. There are many layout patterns that obey $0.8\mu m$ rules. These patterns also include many sets of leaf cell layouts that must be pitchmatched. We used the pitchmatching program to convert those layouts to new ones that obey $0.5\mu m$ rules.

Fig. 9 shows an example of pitchmatching. Three kinds of similar leaf cells are pitchmatched so that the y-coordinates of the ports on the left- and right-hand sides of their boundaries can be made equal. Each leaf cell includes 50 transistors and 16 or 17 ports on the left- and right-hand sides of the boundary. Only the horizontal wires of metal 1 layer that have ports at left- or right-endpoint have grid constraints. Since the ratio α of the number of vertices having grid constraints to that of the total vertices is not so large, we applied pitchmatching algorithm **EXTENT** to the layout. The processing time for generating the **Core Problem** was about 100 minutes. It should be noted that the processing time for generating the constraint graph is excluded. The constraint graph of the **Core Problem**, which has about 100 vertices, was solved in about ten seconds of processing time. Finally, the **Equivalent Problem** was solved in about ten minutes. These processing times were measured by a 15 MIPS computer.

VI. Conclusions

We have described an algorithm for pitchmatching cell layouts with multiple grid constraints. Extending the method for extracting the core problem from layouts having no grid constraints, we devised a framework for pitchmatching layouts having multiple grid constraints. In addition, a sufficient condition for successful application of the pitchmatching algorithm was presented. The computational complexity of the algorithm was compared with a naive approach. On the basis of the comparison, we clarified the condition to make the proposed algorithm more efficient than the naive approach. Finally, we presented two examples and a result of application to a realistic

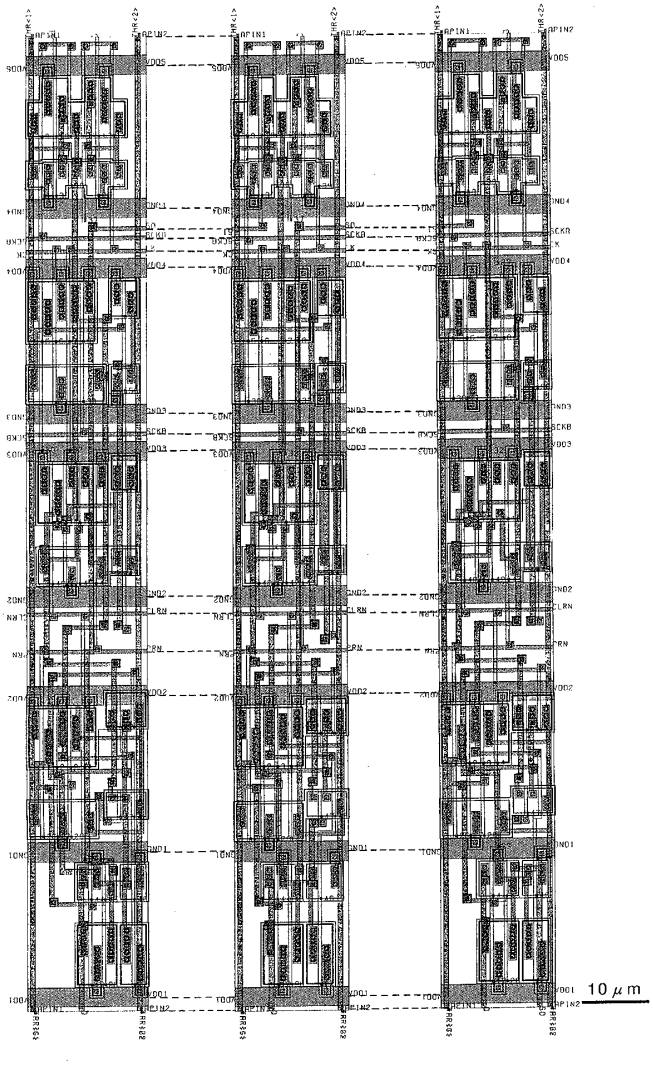

Fig. 9. An example of pitchmatching. Three leaf cells having 50 transistors are pitchmatched with the **EXTENT** algorithm.

layout.

Acknowledgements

The encouragement and support provided by Tohru Adachi and Hitoshi Kitazawa is greatly appreciated. Discussions with Akira Onozawa and Ritsu Kusaba were very helpful.

References

[1] C.S. Bamji and R. Varadarajan, "Hierarchical pitch-matching compaction using minimum design," in *Proc. of ACM/IEEE 28th Design Automation Conf.*, June 1992, pp. 311-317.

[2] J. Dao, N. Matsumoto, T. Hamai, C. Ogawa, and S. Mori, "A compaction method for full chip VLSI layouts," in *Proc. of ACM/IEEE 29th Design Automation Conf.*, June 1993, pp. 407-412.

[3] M.Y. Hsueh and D.O. Pederson, " Computer-aided layout of LSI circuit building-blocks," in *Proc. Int. Symp. Circuits Syst.*, July 1979, pp. 474-477.

[4] J-F. Lee, "VLSI layout compaction with grid and mixed constraints," *IEEE Trans. Computer-Aided Design*, vol. 6, no. 5, pp. 903-910, Sept. 1987.

[5] J-F. Lee, "A layout compaction algorithm with multiple grid constraints," in *Proc. Int. Conf. Computer Design*, 1991, pp. 30-33.

[6] J-F. Lee and D.T. Tang, "HIMALAYAS-A hierarchical compaction system with a minimized constraint set," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1992, pp. 150-157.

[7] C.E. Leiserson and J.B. Saxe, "A mixed-integer linear programming problem which is efficiently solvable," *Journal of Algorithms*, vol. 9, pp. 114-128, 1988.

[8] Y.Z. Liao and C.K. Wong, "An algorithm to compact a VLSI symbolic layout with mixed constraints," *IEEE Trans. Computer-Aided Design*, vol. 2, no. 2, pp. 62-69, Apr. 1983.

[9] R. Okuda, T. Sato, H. Onodera and K. Tamaru, "An efficient algorithm for layout compaction problem with symmetry constraints," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1989, pp. 148-151.

[10] R. Okuda, T. Sato, H. Onodera, and K. Tamaru, "An algorithm for layout compaction problem with symmetry constraints," *Trans. of IEICE*, vol. J73-A, no. 3, pp. 536-543, March 1990.

[11] T. Yoshimura, "A graph theoretical compaction algorithm," in *Proc. Int. Symp. Circuits Syst.*, 1985, pp. 1455-1458.