Chapter 17: Analytical Methods in VLSI Placement

Ulrich Brenner* Jens Vygen*

Abstract

Placement becomes easy if we allow the cells to overlap. Analytical placement works with this relaxation, minimizing a function that estimates netlength. Overlaps are then removed successively by partitioning the chip area and the set of cells, and assigning the cells to smaller and smaller regions. The first successful analytical placers combined quadratic placement with bipartitioning. Later, algorithms for geometric quadrisection and multisection have been found that scale equally well and lead to better solutions. This approach is used intensively in industry.

*Research Institute for Discrete Mathematics, University of Bonn, Lennéstr. 2, 53113 Bonn, Germany

1 Introduction

The basic idea of analytical placement consists of first placing the cells optimally in terms of an appropriate netlength estimation (but without considering disjointness constraints) and then working towards disjointnesss. For the second step, we can distinguish two main approaches. One method consists of modifying the objective function in small steps in order to force cells to move away from each other. Such force-directed approaches will be described in Chapter 18. In this chapter, we will consider methods that reduce overlaps by recursive partitioning of the chip area and the set of cells to be placed. This partitioning is done in such a way that no subregion of the chip area contains more cells than fit into it. Consequently, when the regions are small enough, the cells will be spread over the chip area.

Such an analytical placer is illustrated in Figure 1. The large grey objects are preplaced macros. The first picture shows a placement of the movable cells (red) with minimum squared netlength (with many overlaps). Then, in each partitioning step, the regions and the sets of cells are divided into four parts, colored red, blue, yellow and green. We will explain the details later in this chapter.

[Figure 1 about here.]

Analytical placement is based on the ability to minimize netlength ef-

ficiently. Therefore we first discuss this in Section 2. We define various measures for netlength and show how to minimize linear and quadratic netlength. For reasons that we will discuss, most analytical placers use quadratic netlength. Important properties of placements with minimum quadratic netlength are summarized in Section 3.

Minimizing quadratic netlength goes back to Tutte [1963] who used it for finding straight-line embeddings of planar graphs. Then, this technique has been applied to VLSI placement by Fisk, Caskey and West [1967], Quinn [1975], and Quinn and Breuer [1979]. They tried to reduce overlaps between cells by computing iteratively repulsing forces (see Chapter 18).

Probably the first approach to combine algorithms for minimizing netlength with recursive partitioning has been presented by Wipfler, Wiesel and Mlynski [1982]. They adapt the approach by Quinn and Breuer [1979] and used the result as a guideline for recursive bisection steps. We explain bisection and the more sophisticated approaches used today in Section 4.

In Section 5 we describe methods how the partitioning results can be incorporated in the ensuing netlength optimization steps. Section 6 deals with practical aspects of analytical placement implementations.

2 How to Minimize Netlength

2.1 What is Netlength?

As discussed in Chapter 14, it is not easy to say what a good placement is. The main design objectives timing, power consumption, and manufacturing cost can be influenced only indirectly by placement, as later design steps such as timing optimization or routing follow. Nevertheless there is a need for objective functions that can be evaluated fast.

The most widely adopted quality measure is netlength. Netlength can be defined in various ways, but the idea is always to estimate the wirelength after routing a given placement. Timing is typically taken into account by giving critical nets a higher weight (see Chapter 21).

In order to allow for fast estimation (and possibly optimization) of wirelength, one considers each net individually. This assumes that each net can be wired optimally, disregarding other nets. Of course this is not the case, but it is a reasonable approximation, at least for the majority of the nets and in particular for the most critical ones, unless there is serious routing congestion (which one should avoid anyway; cf. Chapter 22).

For each net we can consider a shortest rectilinear Steiner tree connecting the pins (see Chapter 24), but we shall also consider other estimates. Formally we define:

Definition 2.1 *Given a set $\mathcal{N}$ of disjoint* nets*, each of which is a set of* pins*,* net weights $w : \mathcal{N} \to \mathbb{R}_{\geq 0}$*, pin positions* $(x, y) : \bigcup \mathcal{N} \to \mathbb{R}^2$*, and a function* $\mathcal{M} : \{V \subseteq \mathbb{R}^2 \mid 2 \leq |V| < \infty\} \to \mathbb{R}_{\geq 0}$ *(a* net model*), the (weighted)* netlength with respect to $\mathcal{M}$ *is*

$$\sum_{N \in \mathcal{N}} w(N) \mathcal{M}(\{(x, y)(p) : p \in N\}).$$

Typically a pin shape consists of several rectangles, but this is largely ignored during placement, and a representative point is chosen for each pin. As pin shapes are relatively small, the error resulting from this simplification is also rather small, at least in global placement. Detailed placement (legalization; cf. Chapter 20) can improve by considering the actual pin shapes.

The most natural net model, which is closest to the actual wirelength to be expected after routing, is the minimum length of a rectilinear Steiner tree.

However, computing a shortest rectilinear Steiner tree for a given set of points in the plane is *NP*-hard (Garey and Johnson [1977]). This is one reason why other net models are useful. The following net models have been considered in placement (see also Chapters 7 and 14). Let $V \subseteq \mathbb{R}^2$ be a finite set of points in the plane.

- STEINER(V) is the length of a shortest rectilinear Steiner tree for V.

- BB(V) is half the perimeter of the *bounding box* of V, i.e.

$$\max_{(x,y)\in V} x - \min_{(x,y)\in V} x + \max_{(x,y)\in V} y - \min_{(x,y)\in V} y.$$

- CLIQUE(V) is $\frac{1}{|V|-1}$ times the sum of rectilinear distances over all pairs of points in V, i.e.

$$\frac{1}{|V|-1} \sum_{(x,y),(x',y')\in V} (|x-x'| + |y-y'|).$$

- STAR(V) is the minimum total rectilinear distance of an auxiliary point to all elements of V, i.e.

$$\min_{(x',y')\in\mathbb{R}^2} \sum_{(x,y)\in V} (|x-x'| + |y-y'|).$$

The factor $\frac{1}{|V|-1}$ in the clique estimate is standard in order to avoid that nets with many pins dominate the netlength, but other factors (like $\frac{2}{|V|}$) have also been used (see e.g. Alpert, Kahng and Yao [1999]).

The bounding box and the star estimate can both be determined in linear time: the auxiliary point for a star can be found by two median searches (Blum et al. [1973]). The clique estimate can be computed in $O(|V| \log |V|)$ time by scanning the points after sorting in each coordinate.

The following result tells how well the other three net models approximate the length of an optimum rectilinear Steiner tree. For two-terminal nets all the net models are identical.

[Table 1 about here.]

Theorem 2.2 *Let V be a finite set of points in $\mathbb{R}^2$ and $n := |V| \geq 3$. Then Table 1 shows an upper bound on $\frac{\mathcal{M}_1(V)}{\mathcal{M}_2(V)}$ for net models $\mathcal{M}_1$ (row) and $\mathcal{M}_2$ (column) from* BB, STEINER, CLIQUE, STAR.

As an example how to read the table, the entry in the second row and third column says that $\textsc{steiner}(V) \leq \frac{9}{8}\,\textsc{clique}(V)$ for all V and $\textsc{steiner}(V) \leq \textsc{clique}(V)$ if $n \neq 4$. All inequalities are essentially tight for all n. This result is due to Brenner and Vygen [2001].

In particular, Theorem 2.2 yields $\textsc{steiner}(V) \leq \textsc{clique}(V) \leq \textsc{star}(V)$ for $n \neq 4$. Hence the clique model is superior to the star model as it estimates the length of an optimum rectilinear Steiner tree more accurately. Indeed, a clique is an optimum graph with fixed topology in this respect:

Theorem 2.3 *Let $n \in \mathbb{N}$, $n \geq 2$. Let G be a connected undirected graph with $V(G) \supseteq \{1, \ldots, n\}$, and with edge weights $w : E(G) \to \mathbb{R}_{>0}$.*
For $x, y : \{1, \ldots, n\} \to \mathbb{R}$ let

$$\mathcal{M}_{(G,w)}(x,y) := \min\Bigg\{ \sum_{e=\{u,v\}\in E(G)} w(e)(|x(u)-x(v)| + |y(u)-y(v)|) \;\Big|\; x, y : V(G) \setminus \{1,\ldots,n\} \to \mathbb{R} \Bigg\}.$$

Now define $r(G,w)$ to be the ratio of supremum over infimum of the set

$$\Big\{ \mathcal{M}_{(G,w)}(x,y) \quad | \quad x, y : \{1,\ldots,n\} \to \mathbb{R},\ \textsc{steiner}(\{(x(1),y(1)),\ldots,(x(n),y(n))\}) = 1 \Big\}.$$

Then this ratio is minimum for the complete graph on $\{1, \ldots, n\}$ *with uniform weights; it equals* $\frac{3}{2}$ *for* $n = 4$ *and* $\frac{\lceil \frac{n}{2} \rceil \lfloor \frac{n}{2} \rfloor}{n-1}$ *for* $n \neq 4$*. In this sense, the clique model is optimal for all* n*.*

This is also a result of Brenner and Vygen [2001]. A special case that is interesting in the context of net models for min-cut approaches was considered before by Chaudhuri et al. [2000].

How fast a net model can be computed and how good it approximates the shortest rectilinear Steiner tree are not the only criteria for net models. Another important issue is how well we can optimize netlength with respect to a given net model, assuming that we do not care about overlaps. This will be discussed in the following sections.

2.2 Minimizing Netlength

A key step in analytical placement is to find a placement that minimizes netlength (with respect to a certain net model), disregarding overlaps. This step assumes that there are some fixed pins, since otherwise one can achieve netlength close to zero by placing everything on the same position.

The netlength depends on the pin positions. Each pin either belongs to a movable cell or has a fixed position. We write $\gamma(p)$ to denote the cell that p belongs to, and $\gamma(p) := \square$ if p is fixed. We denote by $(x_{\text{offs}}(p), y_{\text{offs}}(p))$ the

offset of p with respect to $\gamma(p)$, or the absolute position of p if p is fixed.

A *placement* is a pair of coordinates $(x(c), y(c))$ for each $c \in C := \{\gamma(p) \mid p \in P\} \setminus \{\square\}$. It implies pin positions $(x(p), y(p)) = (x(\gamma(p)) + x_{\text{offs}}(p), y(\gamma(p)) + y_{\text{offs}}(p))$ for all $p \in P$, where $x(\square) := 0$ and $y(\square) := 0$.

Thus, for a given net model $\mathcal{M}$, and given net weights $w : \mathcal{N} \to \mathbb{R}_{>0}$, minimizing netlength is the problem of finding a placement minimizing $\sum_{N \in \mathcal{N}} w(N)\mathcal{M}(\{(x(\gamma(p)) + x_{\text{offs}}(p), y(\gamma(p)) + y_{\text{offs}}(p)) : p \in N\})$. Let us stress once more that we do not care about overlaps here.

Many net models are the sum of two independent parts, one depending on x-coordinates only, and the other one depending on y-coordinates only. Examples are BB, CLIQUE, and STAR, but also common quadratic models (see Section 2.4). For such net models, x- and y-coordinates can be optimized separately. This results in two independent "one-dimensional" problems.

2.3 How to Minimize Linear Netlength

Netlength with respect to any of the net models BB, CLIQUE, or STAR, can be minimized efficiently (if we do not care about overlaps). As discussed above, the coordinates can be considered separately, and we use only x-coordinates in our exposition.

The problem of minimizing weighted bounding box netlength can be writ-

ten as a linear program (LP) by introducing two variables l_N and r_N for the leftmost and rightmost coordinate of a pin of each net N (i.e. the edges of the bounding box), and writing

$$\min \sum_{N \in \mathcal{N}} w(N)(r_N - l_N)$$

subject to

$$l_N \leq x(\gamma(p)) + x_{\text{offs}}(p) \leq r_N$$

$$\text{for all } p \in N \in \mathcal{N}.$$

This is an LP with $2|\mathcal{N}| + |C|$ variables and $2|P| + |C|$ linear inequality constraints. Fortunately one does not have to use generic LP solvers but can exploit the special structure of this linear program. As noted first by Cabot, Francis and Stary [1970], this LP is the dual of a transshipment problem (uncapacitated minimum cost flow problem), with a vertex for each variable and two arcs for each pin. More precisely, let G be the digraph with vertex set $V(G) := \{l_N, r_N \mid N \in \mathcal{N}\} \cup C \cup \{\square\}$ and arc set $E(G) := \{(l_N, \gamma(p)), (\gamma(p), r_N) \mid p \in N \in \mathcal{N}\}$. The cost of an arc $(l_N, \gamma(p))$ is $x_{\text{offs}}(p)$, and the cost of $(\gamma(p), r_N)$ is $-x_{\text{offs}}(p)$. Then we look for a minimum cost flow carrying one unit out of l_N and one unit into r_N for each $N \in \mathcal{N}$.

Given a minimum cost flow, it is easy to obtain an optimum dual solution (a feasible potential in the residual graph) by a shortest path computation.

The theoretically fastest known algorithm for transshipment problems,

due to Orlin [1993], has a running time of $O(n \log n(m + n \log n))$, where n is the number of vertices and m is the number of arcs. In our case, we have $n = |C| + 2|\mathcal{N}|$ and $m = 2|P|$. With the realistic assumption $|\mathcal{N}| \geq |\mathcal{C}|$ we get a running time of $O(|\mathcal{N}| \log |\mathcal{N}|(|P| + |\mathcal{N}| \log |\mathcal{N}|))$. See Korte and Vygen [2008] for more details on minimum cost flows.

The star and the clique model (and any other linear model with fixed topology in the sense of Theorem 2.3) can be reduced to the bounding box model by adding a cell with a single pin for each auxiliary point and replacing each net equivalently by an appropriate set of two-terminal nets. Of course, this may increase the number of nets substantially.

The converse is also true: optimizing the bounding box netlength as above is equivalent to minimizing netlength in a certain netlist containing two-terminal pins only, computable as follows. Introduce fixed pins at the leftmost possible position L and at the rightmost possible position R. Moreover, introduce cells l_N and r_N, each with a single pin, for each net N, and replace the net N by $2|P| + 2$ two-terminal nets, one connecting L and l_N with weight $w(N)(|N| - 1)$, another one connecting r_N and R with weight $w(N)(|N| - 1)$, and for each pin $p \in N$ a net connecting l_N and p and a net connecting p and r_N, each of weight $w(N)$. For any placement of the pins, the weighted netlength of the new netlist is

$\sum_{N\in\mathcal{N}} w(N)((|N|-1)(|R-x(r_N)|+|x(l_N)-L|)+\sum_{P\in N}(|x(p)-x(l_N)|+|x(r_N)-x(p)|))$. For a solution minimizing this expression we have $x(l_N)=\min_{p\in N} x(p)$ and $x(r_N)=\max_{p\in N} x(p)$, and the above expression reduces to $\sum_{N\in\mathcal{N}} w(N)((|N|-1)(R-L)+(x(r_N)-x(l_N)))$. Except for a constant additive term this is the weighted bounding box netlength.

For netlists with two-terminal nets only and zero pin offsets, an instance is essentially an undirected graph G with edge weights w, a subset $C\subset V(G)$ of movable vertices and coordinates $x(v)$ for $v\in V(G)\setminus C$. Minimizing bounding box netlength then means finding coordinates $x(c)$ for $c\in C$ such that $\sum_{e=(v,w)\in E(G)} w(e)|x(v)-x(w)|$ is minimized. For this special case, Picard and Ratliff [1978] and later also Cheung [1980] proposed an alternative solution, which may be faster than the minimum cost flow approach described above. Their algorithms solve $|V(G)\setminus C|-1$ minimum s-t-cut problems in an auxiliary digraph with at most $|C|+2$ vertices (including s and t) and at most $|E(G)|+|C|$ arcs. Finding a minimum s-t-cut can be accomplished by any maximum flow algorithm. In a digraph with n vertices and m edges the theoretically fastest one, due to King, Rao and Tarjan [1994], runs in $O(nm\log_{2+m/(n\log n)} n)$ time. This approach may be faster than any transshipment algorithm in some cases, in particular if there are only few fixed pin positions and significantly more two-terminal nets than cells. However,

it is unclear whether nonzero pin offsets can be incorporated.

2.4 How to Minimize Quadratic Netlength

Quadratic netlength is a widely-used objective function in analytical placement (see Kleinhans et al. [1991] (GORDIAN), Alpert et al. [1997], Vygen [1997] (BONNPLACE)). It is also in use as a starting point for many force-directed approaches (see Chapter 18)

For quadratic optimization any net model which replaces each net by a graph with fixed topology may be applied. We will describe quadratic netlength optimization for CLIQUE, the generalization to other graphs is straightforward. Since x- and y-coordinates can be computed independently, we again restrict our description to x-coordinates. We ask for x-coordinates $x(c)$ for each $c \in C$ minimizing

$$\sum_{N \in \mathcal{N}} \frac{w(N)}{|N|-1} \sum_{p,q \in N} \Big((x(\gamma(p)) + x_{\text{offs}}(p)) - (x(\gamma(q)) + x_{\text{offs}}(q)) \Big)^2.$$

Thus, up to constant terms the objective function is

$$\sum_{N \in \mathcal{N}} \frac{w(N)}{|N|-1} \sum_{p,q \in N} \Big[x(\gamma(p)) \Big(x(\gamma(p)) + 2x_{\text{offs}}(p) - x(\gamma(q)) - 2x_{\text{offs}}(q) \Big) + x(\gamma(q)) \Big(x(\gamma(q)) + 2x_{\text{offs}}(q) - x(\gamma(p)) - 2x_{\text{offs}}(p) \Big) \Big].$$

Minimizing this function is equivalent to solving the quadratic program (QP)

$$\min_x x^T A x - 2b^T x, \tag{1}$$

where $A = (a_{c_1,c_2})_{c_1,c_2 \in C}$ and $b = (b_c)_{c \in C}$ with

$$a_{c_1,c_2} := \begin{cases} \sum_{N \in \mathcal{N}} \sum_{\substack{p,q \in N: \\ \gamma(p)=c_1, \gamma(q) \neq c_1}} \frac{w(N)}{|N|-1} & : \quad c_1 = c_2 \\ \sum_{N \in \mathcal{N}} \sum_{\substack{p,q \in N: \\ \gamma(p)=c_1, \gamma(q)=c_2}} -\frac{w(N)}{|N|-1} & : \quad c_1 \neq c_2 \end{cases}$$

and

$$b_c := \sum_{N \in \mathcal{N}} \sum_{\substack{p,q \in N: \\ \gamma(p)=c, \gamma(q) \neq c}} \frac{w(N)}{|N|-1} (x_{\text{offs}}(q) - x_{\text{offs}}(p)).$$

Here, the notation x^T denotes transposition of x.

If the netlist is connected, then the matrix A is positive definite, and the function $x \mapsto x^T Ax - 2b^T x$ is convex and has a unique minimum x, namely the solution of the linear equation system $Ax = b$. Moreover, the matrix A is sparse since the number of non-zero entries is linear in the number of pins. With these additional properties, (1) can be solved efficiently, for example, by the conjugate gradient method (Hestenes and Stiefel [1952]).

We describe its idea for minimizing $f(x) = x^T Ax - 2b^T x$. The algorithm starts with an initial vector x_0. In each iteration i ($i = 1, 2, \ldots$) we choose a direction d_i and a number $t_i \in \mathbb{R}_{\geq 0}$ such that $f(x_{i-1} + t_i d_i) = \min\{f(x_{i-1} + td_i) \mid t \in \mathbb{R}\}$, so we have to solve a one-dimensional quadratic optimization problem to compute t_i. Then, we set $x_i := x_{i-1} + t_i d_i$. For iteration 1, we just set $d_1 := -\nabla f(x_0) = -2Ax_0 + 2b$, i.e. we search for a minimum in the direction of the gradient. Obviously, we have $d_1^T \nabla f(x_0 + t_1 d_1) = d_1^T \nabla f(x_1) = 0$. The idea of the conjugate gradient method is to choose the

directions d_i in such a way that we have in each iteration i: $d_j^T \nabla f(x_i) = 0$ for all $j \in \{1, \ldots, i\}$. This will be the case if all directions are A-conjugate, i.e. if they are non-zero and if for all pairs of directions d_j, d_i we have $d_j^T A d_i = 0$. Then, since the search directions are linearly independent, the gradient $\nabla f(x_i)$ will be 0 after at most n iterations because it is orthogonal to n linearly independent vectors in $\mathbb{R}^n$. The A-conjugacy of the search vectors can be achieved by setting $d_i := -\nabla f(x_i) + \alpha_i d_{i-1}$ for an appropriate value of $\alpha_i \in \mathbb{R}$.

In each iteration of the conjugate gradient method, one multiplication of the $n \times n$-matrix A and an n-dimensional vector are necessary. The number of iterations is bounded by n, but in practice much less iterations are necessary. Generally, if x^* is the optimum solution of (1), we have for $i \in \mathbb{N}$:

$$||x_{i+1} - x^*||_A \leq \frac{\text{cond}_2(A) - 1}{\text{cond}_2(A) + 1} ||x_i - x^*||_A$$

where $||x||_A = \sqrt{x^T A x}$ and $\text{cond}_2(A) := ||A||_2 \cdot ||A^{-1}||_2$ (with $||A||_2 := (\sum_{c_1 \in C} \sum_{c_2 \in C} a_{c_1,c_2}^2)^{\frac{1}{2}}$). In other words: the difference between the vectors x_i and the optimum solution decreases exponentially, and the smaller the condition $\text{cond}_2(A)$ of matrix A is, the faster the algorithm converges. Thus, often pre-conditioning methods are applied to matrix A that reduce the condition. Note that such a pre-conditioning only makes sense for our problems if the resulting matrix is still sparse.

According to Theorem 2.3, CLIQUE is the most accurate approximation of a rectilinear Steiner tree among the net models with fixed topology, and it seems to be reasonable to use this model even when minimizing quadratic netlength. However, for quadratic netlength, CLIQUE may be replaced equivalently by STAR. Indeed one can easily show that replacing a clique of n pins with uniform edge weights w by a star with uniform weights nw does not change the optimum; this will reduce memory consumption and running time when applied to cliques exceeding a certain cardinality.

[Figure 2 about here.]

2.5 Examples

For analytical placers the existence of some preplaced pins is mandatory. Without preplaced pins all cells would be placed at almost the same position in the global optimization (with any reasonable objective function), so we would not get any useful information. Input/output pins (I/O pins) of the chip will usually be preplaced, and often some of the larger macros will be placed and fixed before placement.

The connections to preplaced pins help to pull cells away from each other but their effect is different for quadratic and linear netlength. This is illustrated for three chips in Figure 2. The first two chips contain some preplaced

macros (grey) while in the third one, only the I/O pins are fixed and all cells are movable. For each chip, we present two optimum placements for the movable cells (red) minimizing either linear or quadratic netlength. Obviously, in quadratic placement, the connections to the preplaced pins are able to pull the movable cells away from each other while with the linear objective function, the cells are concentrated at only a very small number of different locations.

2.6 Other Objective Functions

Though most analytical placement algorithms optimize quadratic netlength, there are some approaches that use different objective functions. Most of them try to approximate linear netlength by smooth differentiable functions.

The objective functions that we consider in this section consist again of a part for the x-coordinate and a part for the y-coordinate that can be computed independently. We will present again only the part for the x-coordinate.

Sigl, Doll and Johannes [1991] (GordianL) try to combine advantages of linear and quadratic netlength optimization. Applying the star model, they minimize quadratic netlength but approximate linear netlength by setting net weights that are reciprocally proportional to an estimation of the linear

netlength. More precisely, they iteratively compute sequences of locations $(x_i(p), y_i(p))$ $(i = 0, 1, \ldots)$ for all pins p, and in iteration $i+1$ they estimate the length of a net N by

$$\frac{\sum_{p \in N} \left(x_{i+1}(p) - \frac{1}{|N|} \sum_{q \in N} x_{i+1}(q) \right)^2}{\sum_{p \in N} \left| x_i(p) - \frac{1}{|N|} \sum_{q \in N} x_i(q) \right|}.$$

They stop as soon as the locations of the pins do not change significantly anymore. However, there is no proof of convergence.

The single iterations can be performed quite efficiently but since the computations have to be repeated several times, this method is more time-consuming than just minimizing quadratic netlength. In the experiments presented by Sigl, Doll and Johannes [1991], the running time of GordianL is about a factor of five larger than the running time of Gordian (but GordianL produces better results).

Alpert et al. [1998] approximate the linear netlength $\sum_{p,q \in N} |x(p) - x(q)|$ of a net N by the so-called *β-regularization* (for $\beta > 0$):

$$\text{clique}^x_\beta(N) = \sum_{p,q \in N} \sqrt{((x(p) - x(q))^2 + \beta)}.$$

$\text{clique}^x_\beta(N)$ is obviously differentiable and an upper bound of $\sum_{p,q \in N} |x(p) - x(q)|$. Moreover, we have $\text{clique}^x_\beta(N) \to \sum_{p,q \in N} |x(p) - x(q)|$ for $\beta \to 0$. Of course, net models using other graphs with fixed topology than cliques can

be handled analogously. Alpert et al. [1998] apply the Primal-Dual Newton method that converges to the optimum of this convex objective function.

Kennings and Markov [2002] present a differentiable approximation of the bounding-box netlength. For a net N and parameters $\beta > 0$ and $\eta > 0$, they use

$$\text{BB}^x_{\beta,\eta}(N) = \left(\sum_{p,q \in N} |x(p) - x(q)|^\eta + \beta \right)^{\frac{1}{\eta}}.$$

We have $\text{BB}^x_{\beta,\eta}(N) + \text{BB}^y_{\beta,\eta}(N) \geq \text{BB}(N)$ and $\lim_{\eta\to\infty} \lim_{\beta\to 0} (\text{BB}^x_{\beta,\eta}(N) + \text{BB}^y_{\beta,\eta}(N)) = \text{BB}(N)$.

This function if strictly convex (if each connected component of the netlist contains a preplaced pin) and hence can be optimized by the Newton method.

Kahng and Wang [2004] (APLACE) and Chan, Cong and Sze [2005] (MPL) propose to minimize a differentiable approximation to the bounding-box netlength. For a parameter α, they define

$$\text{BB}^x_\alpha(V) := \alpha \left(\ln \left(\sum_{p \in V} e^{\frac{x(p)}{\alpha}} \right) + \ln \left(\sum_{p \in V} e^{\frac{-x(p)}{\alpha}} \right) \right).$$

It is easy to see that $\text{BB}^x_\alpha(V) + \text{BB}^y_\alpha(V) \to \text{BB}(V)$ for $\alpha \to 0$. Kahng and Wang [2004] combine this function with a smooth potential function that penalizes placement overlaps to a differentiable objective function that they try to optimize by a conjugate gradient method. However, the resulting objective function is not convex anymore. Moreover, the authors do not show if this method converges to any local minimum. For a more detailed description of

the approach, we refer to Chapter 18.

3 Properties of Quadratic Placement

3.1 Relation to Electrical Networks and Random Walks

Quadratic placement has a very nice interpretation in terms of random walks. For our exposition we assume the simplest case that all pin offsets are zero.

Proposition 3.1 *Given a netlist with zero pin offsets, we define a weighted graph as follows: The vertices are the movable objects (cells) and the fixed pins. For each net N and each pair of pins $p, q \in N$ belonging to different cells c, c' we have an edge with endpoints $\{c, c'\}$ and weight $\frac{w(N)}{|N|-1}$. For each net N and each pair of pins $p, q \in N$, where p belongs to cell c and q is fixed, we have an edge with endpoints $\{c, q\}$ and weight $\frac{w(N)}{|N|-1}$. We assume that some fixed pin is reachable from each cell in this graph.*

We consider random walks in this graph. We always start at a cell, and we stop as soon as we reach a fixed pin. Each step consists of moving to a randomly chosen neighbour, where the probabilities are proportional to the edge weights.

For each cell c, let x_c be the expectation of the x-coordinate of the fixed pin where a random walk started in c ends. Then x_c is precisely the position

of c in the quadratic placement.

Proof: It is easy to see that the numbers x_c satisfy the linear equation system $Ax = b$ defined in Section 2.4. As it has a unique solution, it is equal to the quadratic placement. □

This has been generalized to arbitrary pin offsets by Vygen [2007].

Another interpretation of quadratic placement is in the context of electrical networks. Interpret the graph defined above as an electrical network, where edges correspond to connections whose resistance is inversely proportional to the weight, and where a potential of $x(q)$ is applied to each fixed pin q, where $x(q)$ is its x-coordinate. By Ohm's law, a current of $x(c) - x(c')$ is flowing from c to c', where $x(c)$ is the resulting potential of c in this network. By Kirchhoff's law the numbers x also satisfy the above linear equation system.

3.2 Stability

In practice, the final netlist of a chip is not available until very late in the design process. Of course, results obtained with preliminary netlists should allow conclusions on results for the final netlist. Therefore stability is an essential feature of placement algorithms – it is much more important than obtaining results that are close to optimum. When stable placement al-

gorithms are unavailable, one has to enforce stability, e.g. by employing a hierarchical design style, dividing the chip into parts and fixing the position of each part quite early. Clearly, such an unflexible hierarchical approach entails a great loss in quality.

For precise statements we have to formalize the term stability. This requires answers to two questions: When are two placements similar? And what elementary netlist changes should lead to a similar placement?

We first consider the first question. We are not interested in the relative position of two cells unless they are connected. Moreover, if all pins of a net move by the same distance into the same direction, this does not change anything for this net. Therefore the following discrepancy measure was proposed by Vygen [2007]. Again we restrict to zero pin offsets for a simpler notation.

Definition 3.2 *Let a netlist be given, where $\mathcal{N}$ is the set of its nets and $w : \mathcal{N} \to \mathbb{R}_{\geq 0}$ are net weights. Let two placements be given, and let $(x(p), y(p))$ and $(x'(p), y'(q))$ be the position of pin p w.r.t. the first and second placement, respectively.*

Then the discrepancy *of these two placements is defined to be*

$$\sum_{N \in \mathcal{N}} \frac{w(N)}{|N| - 1} \sum_{p,q \in N} \Big(\big(x(p) - x'(p) - x(q) + x'(q)\big)^2 + \big(y(p) - y'(p) - y(q) + y'(q)\big)^2 \Big).$$

We apply this measure to estimate the effect of small netlist changes on the quadratic placement. The most elementary operation is increasing the weight of a net. As discrepancy is symmetric, this covers also reducing the weight of a net, deleting or inserting a net. Thus arbitrary netlist changes can be composed of this operation.

Theorem 3.3 *Let a netlist be given. We assume that each connected component of the netlist graph contains a fixed pin. Let $(x(p), y(p))$ be the position of pin p in the quadratic placement of this netlist, and let $(x'(p), y'(p))$ be its position in the quadratic placement after increasing the weight of a single net N by δ. Then the discrepancy of the two placements is at most $\delta \frac{\lceil \frac{n}{2} \rceil \lfloor \frac{n}{2} \rfloor}{2(n-1)} (X_N^2 + Y_N^2)$, where $n := |N|$ and*

$$X_N := \max\{x(p) \mid p \in N\} - \min\{x(p) \mid p \in N\},$$

$$Y_N := \max\{y(p) \mid p \in N\} - \min\{y(p) \mid p \in N\}.$$

This and similar results are proved in Vygen [2007]. Roughly speaking, they say that small local changes to a netlist do not change the quadratic placement significantly. In this sense, quadratic placement is stable.

We now argue that other approaches are instable. Even for identical input we can obtain placements with large discrepancy:

Theorem 3.4 *There exists a constant $\alpha > 0$ such that for each even $n \geq 4$ there is a netlist with n cells and the following properties: Each cell has*

width $\frac{1}{n}$ *and height* 1. *The chip area in which the cells must be placed is the unit square. Each cell has 3, 4 or 5 pins. All pin offsets are zero. All nets have two terminals. There are two optimum placements (with respect to netlength), which have discrepancy at least* αn.

As each optimum placement is a possible result of any local search algorithm, such algorithms are instable. Similarly, Vygen [2007] shows the instability of min-cut approaches: there are netlists for which a min-cut approach, depending on a tie-breaking rule at the first cut, can produce placements whose discrepancy is proportional to the number of nets (and thus only a constant factor better than the maximum possible discrepancy). Hence these approaches lack any stability.

This is a reason to favour quadratic placement approaches. Of course, quadratic placements usually contain many overlapping cells, and further steps have to be applied to remove overlaps (cf. Figure 2). So far, nobody has succeeded to prove stability of an overall algorithm which produces a feasible placement for any netlist. But at least the basic ingredient, quadratic placement, is stable. Analytical placement algorithms like the ones described in the following, as well as force directed placement approaches like Eisenmann and Johannes [1998] (cf. Chapter 18) try to modify this placement as little as possible while removing overlaps.

4 Geometric Partitioning

4.1 Objectives

After minimizing quadratic (or linear) netlength without considering any disjointness constraints, analytical placers start to remove overlaps by partitioning the chip area into regions and by assigning cells to regions such that no region contains more cells than fit into it. As we have a well-optimized placement (but with overlaps), it seems to be reasonable to change it as little as possible, i.e. to minimize the total distance that cells move.

More formally, the following problems has to be solved. We are given a set C of movable cells and a set R of regions. Each cell $c \in C$ has a size, denoted by $\text{size}(c)$, and each region $r \in R$ has a capacity, denoted by $\text{cap}(r)$. Moreover, for each pair $(c, r) \in C \times R$ we know the cost $d((c, r))$ of moving cell c to region r. The task is to find a mapping $g : C \to R$ such that $\sum_{c \in C : g(c) = r} \text{size}(c) \leq \text{cap}(r)$ for all $r \in R$, minimizing $\sum_{c \in C} d((c, g(c)))$.

Unfortunately, to decide if this problem has any feasible solution is NP-complete even if $|R| = 2$ (Karp [1972]). Hence, it is natural to relax the problem by allowing cells to be distributed to different regions. Then we arrive at the following problem:

Fractional Assignment Problem	
Instance:	• Finite sets C and R; • $\text{size} : C \to \mathbb{R}_{>0}$; • $\text{cap} : R \to \mathbb{R}_{>0}$; • $d : C \times R \to \mathbb{R}_{\geq 0}$;
Task:	Find a mapping $h : C \times R \to [0,1]$ with $\sum_{r \in R} h((c,r)) = 1$ for all $c \in C$ and $\sum_{c \in C} h((c,r)) \cdot \text{size}(c) \leq \text{cap}(r)$ for all $r \in R$, minimizing $\sum_{c \in C} \sum_{r \in R} h((c,r)) \cdot d(c,r)$

Considering this fractional version is sufficient because of the following theorem:

Theorem 4.1 *There is always an optimum solution h of the* Fractional Assignment Problem *where the set $\{c \in C \mid \exists r \in R : h((c,r)) \neq \{0,1\}\}$ has at most $|R| - 1$ elements.*

For a proof we refer to Vygen [2005]. If any optimum solution is given, such an "almost integral" optimum solution can be computed efficiently.

4.2 Bipartitioning

If $|R| = 2$, then the Fractional Assignment Problem is equivalent to the Fractional Knapsack Problem (cf. Korte and Vygen [2008]). The unweighted version of this problem (i.e., $\text{size}(c) = 1$ for all $c \in C$) can

be solved in linear time by using the linear-time algorithm for the MEDIAN PROBLEM described by Blum et al. [1973]. Adolphson and Thomas [1977], Johnson and Mizoguchi [1978], and Balas and Zemel [1980] show how the algorithm for the unweighted version can be used as a subroutine for a linear time algorithm of the FRACTIONAL KNAPSACK PROBLEM with weights (cf. Korte and Vygen [2008] and Vygen [2005]).

Given a non-disjoint placement with minimum (quadratic) netlength, a straightforward partitioning approach consist of bipartitioning the cells set alternately according to the x- and y-coordinates. Indeed, early analytical placement algorithms that have been presented by Wipfler, Wiesel and Mlynski [1982], Cheng and Kuh [1984], Tsay, Kuh and Hsu [1988] (PROUD), and Jackson and Kuh [1989] apply such a method.

Another analytical placement algorithm based on bipartitioning is GORDIAN (Kleinhans et al. [1991]). The authors try to improve the result of a partitioning step by reducing the number of nets that are cut without increasing the cell movement too much. To this end, they vary the capacities of the subregions (within a certain range), compute cell assignments for the different capacity values and keep the one with the smallest cut. Moreover, cells may be interchanged between the two subsets after bipartitioning if this reduces the number of nets that are cut.

[Figure 3 about here.]

Sigl, Doll and Johannes [1991] (GordianL) describe an iterative method for bipartitioning. They cope with the problem that if many cells have very similar locations before a partitioning step, the decision to which subset they are assigned is more or less arbitrary. Their heuristic works in two phases, illustrated in Figure 3. Assume that a set of cells (Figure 3 (a)) has to be divided into two parts and that we ask for a vertical cut. First, cells with very small or very big x-coordinates are assigned to the left or to the right subset of the partition. In Figure 3 (b), the cells that reach out to the left of coordinate x_1 (shown in green) are assigned to the left part, and the cells that reach out to the right of coordinate x_2 (red) are assigned to the right part. The idea is that the assignment of these cells can hardly be wrong and that the connectivity to them should be used when assigning the remaining cells. The pre-assigned cells are forced to move further to the left or to the right depending on their assignment. With these additional constraints new positions for all cells to be partitioned are computed (in GordianL minimizing an approximation of linear netlength, cf. Section 2.6) as shown in Figure 3 (c). Finally, these new positions are used to compute the assignment of the cells (Figure 3 (d)).

4.3 Quadrisection

BonnPlace, an analytical placer proposed by Vygen [1997], makes use of a linear-time algorithm for a special case of the Fractional Assignment Problem. If R consist of four elements r_1, r_2, r_3, and r_4 such that $d((c, r_1)) + d((c, r_3)) = d((c, r_2)) + d((c, r_4))$ for all $c \in C$, then the Fractional Assignment Problem can be solved in time $O(|C|)$ (see Vygen [2005] for a proof). This condition is met if R is the set of the four quadrants of the plane and $d(c, r)$ is the L_1 distance between c and r. Such a partitioning is shown in Figure 4 where the cells are colored according to the region that they are assigned to (e.g. the red cells will go to the upper left quadrant). The borderlines between the cell subsets are horizontal, vertical and diagonal lines that form a geometric structure that is called *American map*. Vygen [2005] proves that an American map corresponding to an optimum partitioning can be computed in linear time. The algorithm can be seen as a two-dimensional generalization of the median algorithm by Blum et al. [1973].

[Figure 4 about here.]

4.4 Grid Warping

Xiu et al. [2004] (see also Xiu and Rutenbar [2005]) start with a placement

that minimizes quadratic netlength but partition the set of cells by borderlines that do not have to be horizontal or vertical.

Assume, for example, that we want to partition the set of cells (and the chip area) into four parts. The chip area is partitioned by a horizontal and a vertical cut running through the whole chip area, thus forming four rectangular regions. In order to partition the set of cells, Xiu et al. [2004] compute a borderline l_1 connecting the upper edge of the chip area to the lower edge and two borderlines l_2 and l_3 connecting the left (right) edge of the chip area to l_1 (see Figure 5).

[Figure 5 about here.]

These three borderlines partition the set of cells into four subsets C_1, C_2, C_3, and C_4, and each subset is assigned in the obvious way to a subregion.

The borderlines used to partition the set of cells shall be chosen such that capacity constraints are met for the subregions and such that routing congestion and netlength are minimized when the cells are moved to their regions. Since it seems to be hard to find optimal cutlines with these optimization goals, the authors apply local search to compute the borderlines. They argue that this is good enough as the number of variables is small (two variables for each cutline). As the algorithm does not only use vertical and horizontal cutlines for the partitioning of the cells and warps the placement

in a partitioning step, the authors call it "grid-warping" partitioning.

4.5 Multisection

The FRACTIONAL ASSIGNMENT PROBLEM is solvable in polynomial time since it can be seen as a HITCHCOCK TRANSPORTATION PROBLEM, as special version of a minimum-cost flow problem.

An efficient algorithm for the unbalanced instances that occur in placement (where often $|C|$ is much larger than $|R|$) has been proposed by Brenner [2005] who proved the following theorem:

Theorem 4.2 *The* FRACTIONAL ASSIGNMENT PROBLEM *can be solved in time* $O(nk^2(\log n + k \log k))$ *where* $n := |C|$ *and* $k := |R|$.

Thus, for fixed k, the FRACTIONAL ASSIGNMENT PROBLEM can be solved in time $O(n \log n)$. This *multisection* is slower than the linear-time algorithm for quadrisection proposed by Vygen [2005] but the algorithm is more flexible since it can handle an arbitrary number of regions and an arbitrary costs function. This flexibility can be used e.g. for reducing the number of partitioning steps, and for a more intensive local optimization in repartitioning (see Section 6.1 and Brenner and Struzyna [2005]). Moreover, movement costs are not restricted to L_1-distances. For example they could take blocked areas (e.g. used by preplaced macros) into consideration.

An example for multisection with nine regions and L_1-distances as movement costs is shown in Figure 6. Again, the colors of the cells indicate the region that they are assigned to. For example, the red cells will go to the upper left region. As expected, American map structures reappear.

[Figure 6 about here.]

5 How to Use the Partitioning Information

After a partitioning step, each cell is assigned to a region of the chip area. Before the regions (and the corresponding sets of cells) are partitioned further, we have to ensure that the cells are placed (approximately) within their regions. For linear netlength, it is quite obvious how upper and lower bounds on the coordinates of single cells may be added to the LP formulation described in Section 2.3. The LP with such additional constraints is still the dual of a minimum-cost flow problem.

If we want to add linear upper and lower bounds for cell positions to the quadratic program (1), this leads to a quadratic objective function that has to be minimized over a convex set. This problem is solvable in polynomial time, but not efficient enough for large instances. Hence, different approaches are used to take the partitioning information into account. We discuss the

two main techniques in the following.

5.1 Center-of-Gravity Constraints

In order to move each group of cells towards the region that is has been assigned to, Kleinhans et al. [1991] prescribe the center of gravity of each group as the center of the region that this group is assigned to. For each region, this introduces an equation as an additional constraint on the solution of the quadratic program (1). Kleinhans et al. [1991] show how this constrained quadratic program can be reduced elegantly to an unconstrained quadratic program with the following transformation. For n movable cells and k additional constraints, the constrained quadratic program may be written in the form

$$\begin{aligned} \min \quad & x^T A x - 2b^T x \\ \text{s.t.} \quad & (I\ S)x = t \end{aligned}$$

where the $k \times n$-matrix $(I\ S)$ consists of the $k \times k$-identity matrix I and a $k \times (n-k)$-matrix S. With $x = \binom{x_1}{x_2}$ (where $x_1 \in \mathbb{R}^k$ and $x_2 \in \mathbb{R}^{n-k}$), the linear constraints can be written as $x_1 = t - Sx_2$. Hence, we only have to compute the entries of x_2 by solving the following unconstrained problem on $n-k$ variables:

$$\min \binom{t - Sx_2}{x_2}^T A \binom{t - Sx_2}{x_2} - 2b^T \binom{t - Sx_2}{x_2}.$$

By ignoring all constant summands in the objective function, we get the equivalent problem

$$\min x_2^T U^T A U x_2 - 2v^T x_2 \tag{2}$$

where $U := \binom{-S}{I}$ and $v := U^T \left(b - A \binom{t}{0}\right)$. The matrix $U^T AU$ is positive definite if A is positive definite, but usually $U^T AU$ will not be sparse. Therefore, for an efficient solution, an explicit computation of $U^T AU$ must be avoided. Fortunately, the conjugate gradient method (see Section 2.4) only requires to multiply $U^T AU$ with a vector, which can be done by three single multiplications of a sparse matrix and a vector. Hence, provided that the number of constraints is small compared to the number of cells, the conjugate gradient method will efficiently solve problem (2).

Prescribing the centers of gravity of the cell groups is an efficient way to spread the cells over the chip area. However, we cannot be sure that all cells are placed inside their region, which can be a problem for ensuing partitioning steps. Moreover, the constraints may be too strong if we do not demand an even distribution of the cells. If we allow a higher area utilization in some regions, it will often be reasonable to place cells in their region in such a way that their center of gravity is far away from the center of the region.

5.2 Splitting Nets

A second way to reflect the result of partitioning in the quadratic program, proposed by Vygen [1997], consists of splitting nets at the borders of regions. In this approach, we assume that the chip area is partitioned in a grid-like manner by vertical and horizontal cutlines that cross the whole chip.

Suppose that we have bounds $\mu \leq x(c) \leq \nu$ for the x-coordinate of a cell c. For each cell c' that is connected to c but is placed in a window to the right of b (i.e. ν is a lower bound on the x-coordinate of c'), we replace the connection to c' by an artificial connection between c and a fixed pin with x-coordinate ν. Analogously, connections to cells c' that will be placed to the left of μ are replaced by connections to a fixed pin with x-coordinate μ. Connections to fixed pins outside the bounds of a cell are also split.

Note that this splitting is done for x- and y-coordinate independently, so for x-coordinates only the vertical boderlines and for the y-coordinates only the horizontal borderlines between the windows are considered. In particular, in contrast to standard terminal propagation, it is possible (and in fact will happen quite often) that a connection has to be split for the computation of the x-coordinates but not the y-coordinates, and vice versa. This splitting of the nets forces each cell to be placed inside the region that it is assigned to.

However, a problem that has to be addressed in this approach is the following: it may happen that in a region all cells (or most of them) have their external connection to only one direction. In that case a QP solution will place all of them at one border or even in one corner of the region. Such a placement is obviously useless for the next partitioning step based on cell positions. Vygen [1997] proposes to make use of center-of-gravity constraints (see Section 5.1) to modify the placements in these cases. Figure 7 illustrates how this works. The left picture shows the placement with minimum quadratic netlength (splitting connections at the borderlines as described above) without any additional center-of-gravity constraints.

Based on this we compute a new center of gravity for each region in which the current center of gravity of the cells in the region is closer to the border than it would be possible in any disjoint placement. The new center of gravity is (approximately) the closest possible position in a disjoint placement. Then, a new global QP is solved forcing the centers of gravity of the cell groups in these regions to the new prescribed positions. The right-hand side of Figure 7 shows the result. It demonstrates that in particular in the outer regions of the chip area this step changes the placement significantly.

[Figure 7 about here.]

6 Further Techniques

6.1 Repartitioning

In a pure recursive partitioning approach, cells may never leave their regions. However, especially cell assignments in the first paritioning steps may be suboptimal since they are based on placements in which the cell positions may not differ enough. Therefore, there is need for techniques that are able to correct bad decisions in partitioning. Most analytical placers contain some local optimization methods that are executed between the partitioning steps and that allow cells to leave the regions they are assigned to.

In GORDIAN (see Kleinhans et al. [1991]), cells are moved towards their regions by solving a constrained QP (see Section 5.1). As this constrained QP does not force the cells to be placed inside their window, groups of cells that are assigned to different windows may be mixed with each other. In such situations, Kleinhans et al. [1991] reassign cells locally. Let us consider the case when a window is partitioned by a vertical cutline (the case of a horizontal cutline is handled analogously). If after the constrained QP one of the cells assigned to the left window is placed to the right of a cell assigned to the right window, then the two cell subsets are merged and are partitioned once again (using this time the positions of the constrained QP). The old assignment is always replaced by the new assignment. Note that only pairs

of cell groups are considered that belonged to the same window before the previous partitioning, so this re-assignment is the last chance for a cell to leave its window. After all these new assignments have been computed, a new constrained QP is solved. According to the description by Kleinhans et al. [1991] it is not necessary to iterate this method.

In order to allow cells to leave their windows even at a late stage, Vygen [1997] proposes a *repartitioning* technique that tries to find local improvements of the placement. It considers arrays of 2×2-regions (i.e., sets of four regions intersecting in one point) and tries to find a better placement in them. In each such region, the cells are placed with minimum quadratic netlength and are then assigned to the four subregions with a quadrisection step. Finally a local QP is solved where nets are split according to the new assignment. The new placement is accepted if the total netlength has decreased.

This step is done for all 2×2-arrays of regions. This loop is called repeatedly (with different orders of the arrays of regions) as long as it yields a considerable improvement of the weighted netlength.

Repartitioning enables the cells to leave the region in which they are currently placed. It has also been used by Huang and Kahng [1997] in a minimum-cut-based placer and by Xiu and Rutenbar [2005] in their warping

approach.

6.2 Parallelization

Analytical placement methods that use recursive partitioning allow a parallel implementation of most parts of the algorithm. Sometimes, placement and partitioning in one region does not depend on another region, so both regions can be handled in parallel. However, it should be mentioned that many analytical placers apply a global optimization before a partitioning step where all cells are placed simultaneously. For example, in GORDIAN (Kleinhans et al. [1991], the placements with minimum quadratic netlength (with different center-of-gravity constraints) can hardly be parallelized.

Nevertheless, even some parts of these global optimization steps allow a parallel computation if the assignment of the cells to their windows is used as hard constraints. Assume e.g. that we want to compute the x-coordinate of a cell c for which we have the constraints $\mu \leq x(c) \leq \nu$ for some numbers μ and ν. Then, if we minimize linear netlength, the x-coordinate of c can be computed without knowing the x-coordinates of the cell which have to be placed to the left of μ or to the right of ν. Thus, the x-coordinates in different columns given by the regions can be computed in parallel (and analogously for the y-coordinates).

Such a parallel computation is possible as well if quadratic netlength is minimized and connections are split at the borderlines of regions (see Section 5.2).

Also multisection can be done in parallel for separate regions. Moreover, local optimization steps like repartitioning that are often quite time-consuming can be performed efficiently in parallel (see Brenner and Struzyna [2005]).

6.3 Dealing with Macros

Analytical placers can handle cells of different sizes and shapes. However, recursive partitioning has to stop when cells are too big compared to the region size. Hence, for larger macros only a few partitioning steps can be made. Then, macros have to stay more or less at their position.

In GORDIAN (Kleinhans et al. [1991]), a region is only partitioned if it contains a sufficient number of cells, so in the presence of macros the region sizes may differ over a large range at the end of global placement. Finally macros are legalized together with the standard cells.

Other analytical placers such as BONNPLACE (Vygen [1997], Brenner and Struzyna [2005]) place the macros legally as soon as they are too big compared to the region size and fix them before continuing with the recursive

partitioning.

7 Conclusion

Analytical placement is the dominant strategy for VLSI placement today. Decomposing the task into minimizing netlength and partitioning with respect to area constraints is natural. Using quadratic placement and multisection as the two main components has the advantage that both subproblems can be solved almost optimally very efficiently even for the largest netlists. Moreover, this approach has nice stability features and works well in a timing closure framework. Therefore this approach is widely used in industry for many of the hardest placement problems.

References

Adolphson, D.L., and Thomas, G.N. [1977]: A linear time algorithm for a $2 \times n$ transportation problem. SIAM Journal on Computing 6 (1977), 481–486

Alpert, C.J., Chan, T., Huang, D.J.-H., Markov, I., and Yan, K. [1997]: Quadratic placement revisited. Proceedings of the 34th IEEE/ACM Design Automation Conference (1997), 752–757

Alpert, C.J., Chan, T.F., Kahng, A.B., Markov, I.L., and Mulet, P. [1998]: Faster minimization of linear wirelength for global placement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 17 (1998), 3–13

Alpert, C.J., Kahng, A.B., and Yao, S.-Z. [1999]: Spectral partitioning: the more eigenvectors, the better. Discrete Applied Mathematics 90 (1999), 3–26 (DAC 1995)

Balas, E., and Zemel, E. [1980]: An algorithm for large zero-one knapsack problems. Operations Research 28, 1980, 1130–1154.

Blum, M., Floyd, R.W., Pratt, V., Rivest, R.L., and Tarjan, R.E. [1973]: Time bounds for selection. Journal of Computer and System Sciences 7 (1973), 448–461

Brenner, U. [2005]: A faster polynomial algorithm for the unbalanced hitchcock transportation problem. Technical Report 05954 (2005), Research Institute for Discrete Mathematics, University of Bonn

Brenner, U., and Struzyna, M. [2005]: Faster and better global placement by a new transportation algorithm. Proceedings of the 42nd IEEE/ACM Design Automation Conference (2005), 591–596

Brenner, U., and Vygen, J. [2001]: Worst-case ratios of networks in the rectilinear plane. Networks 38 (2001), 126–139

Cabot, A.V., Francis, R.L., and Stary, A.M. [1970]: A network flow solution to a rectilinear distance facility location problem. AIIE Transactions 2 (1970), 132–141

Chan, T.F., Cong, J., and Sze, K. [2005]: Multilevel generalized force-directed method for circuit placement. Proceedings of the IEEE/ACM International Symposium on Physical Design (2005), 227–229

Chaudhuri, S., Subrahmanyam, K.V., Wagner, F., and Zaroliagis, C.D. [2000]: Computing mimicking networks. Algorithmica 26 (2000), 31–49

Cheng, C.-K., and Kuh, E.S. [1984]: Module placement based on resistive network optimization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 3 (1984), 218–225

Cheung, T.-Y. [1980]: Multifacility location problem with rectilinear distance by the minimum-cut approach. ACM Transactions on Mathematical Software 6 (1980), 387–390

Eisenmann, H., and Johannes, F.M. [1998]: Generic global placement and floorplanning. Proceedings of the 35th IEEE/ACM Design Automation Conference (1998), 269–274

Fisk, C.J., Caskey, D.L., and West, L.E. [1967]: ACCEL: automated circuit card etching layout. Proceedings of the IEEE 55 (1967), 1971–1982

Garey, M.R., and Johnson, D.S. [1977]: The rectilinear Steiner tree problem is *NP*-complete. SIAM Journal on Applied Mathematics 32 (1977), 826–834

Hestenes, M.R., and Stiefel, E. [1952]: Methods of conjugate gradients for solving linear systems, Journal of Research of the National Bureau of Standards 49 (1952), 409–439

Huang, D.J.-H., Kahng, A.B. [1997]: Partitioning based standard cell global placement with an exact objective. Proceedings of the IEEE/ACM International Symposium on Physical Design (1997), 18–25

Jackson, M.B., and Kuh, E.S. [1989]: Performance-driven placement of cell-based ICs. Proceedings of the 26th IEEE/ACM Design Automation Conference (1989), 370–375

Johnson, D.B., and Mizoguchi, T. [1978]: Selecting the Kth element in $X+Y$ and $X_1 + X_2 + \ldots + X_m$. SIAM Journal on Computing 7 (1978), 147–153.

Kahng, A.B., and Wang, Q. [2004]: Implementation and extensibility of an analytic placer. Proceedings of the IEEE/ACM International Symposium on Physical Design (2004), 18–25

Karp, R.M. [1972]: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (editors), Complexity of Computer Computations, Plenum Press, New York (1972), 85–103

Kennings, A., and Markov, I. [2002]: Smoothening max-terms and analytical minimization of half-perimeter wirelength. VLSI Design 14 (2002), 229–237

King, V., Rao, S., and Tarjan, R.E. [1994]: A faster deterministic maximum flow algorithm. Journal of Algorithms 17 (1994), 447–474

Kleinhans, J.M., Sigl, G., Johannes, F.M., and Antreich, K.J. [1991]: GORDIAN: VLSI placement by quadratic programming and slicing optimization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 10 (1991), 356–365 (ICCAD 1988)

Korte, B., and Vygen, J. [2008]: Combinatorial Optimization: Theory and Algorithms. Fourth edition. Springer, Berlin 2008

Orlin, J.B. [1993]: A faster strongly polynomial minimum cost flow algorithm. Operations Research 41 (1993), 338–350 (STOC 1988)

Picard, J.C., and Ratliff, H.D. [1978]: A cut approach to the rectilinear distance facility location problem. Operations Research 26 (1978), 422–433

Quinn, N.R. [1975]: The placement problem as viewed from the physics of classical mechanics. Proceedings of the 12th IEEE/ACM Design Automation Conference (1975), 173–178

Quinn, N.R., and Breuer, M.A. [1979]: A force directed component placement procedure for printed circuit boards. IEEE Transactions on Circuits and Systems CAS-26 (1979), 377–388

Sigl, G., Doll, K., and Johannes, F.M. [1991]: Analytical placement: a linear or quadratic objective function? Proceedings of the 28th IEEE/ACM Design Automation Conference (1991), 427–432

Tsay, R.-S., Kuh, E., and Hsu, C.-P. [1988]: Proud: a sea-of-gate placement algorithm. IEEE Design and Test of Computers 5 (1988), 44–56

Tutte, W.T. [1963]: How to draw a graph. Proceedings of the London Mathematical Society 13 (1963), 743–767

Vygen, J. [1997]: Algorithms for large-scale flat placement. Proceedings of the 34th IEEE/ACM Design Automation Conference (1997), 746–751

Vygen, J. [2005]: Geometric quadrisection in linear time, with application to VLSI placement. Discrete Optimization 2 (2005), 362–390

Vygen, J. [2007]: New theoretical results on quadratic placement. Integration, the VLSI Journal 40 (2007), 305–314

Wipfler, G.J., Wiesel, M., and Mlynski, D.A. [1982]: A combined force and cut algorithm for hierarchical VLSI layout. Proceedings of the 19th IEEE/ACM Design Automation Conference (1982), 671–677

Xiu, Z., Ma, J.D., Fowler, S.M., and Rutenbar, R.A. [2004]: Large-scale placement by grid-warping. Proceedings of the 41st IEEE/ACM Design Automation Conference (2004), 351–356

Xiu, Z., and Rutenbar, R.A. [2004]: Timing-driven placement by grid-warping. Proceedings of the 42nd IEEE/ACM Design Automation Conference (2005), 585–590

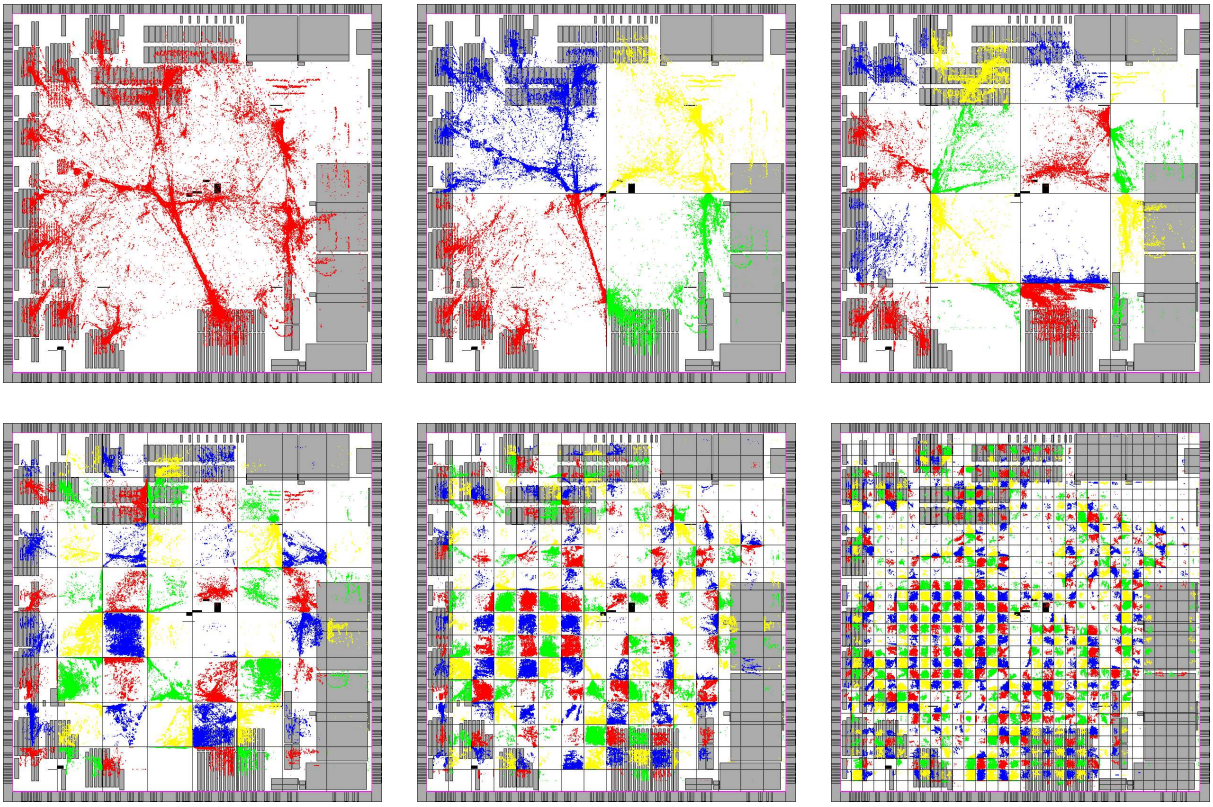

Figure 1: The first six steps of an analytical placer.

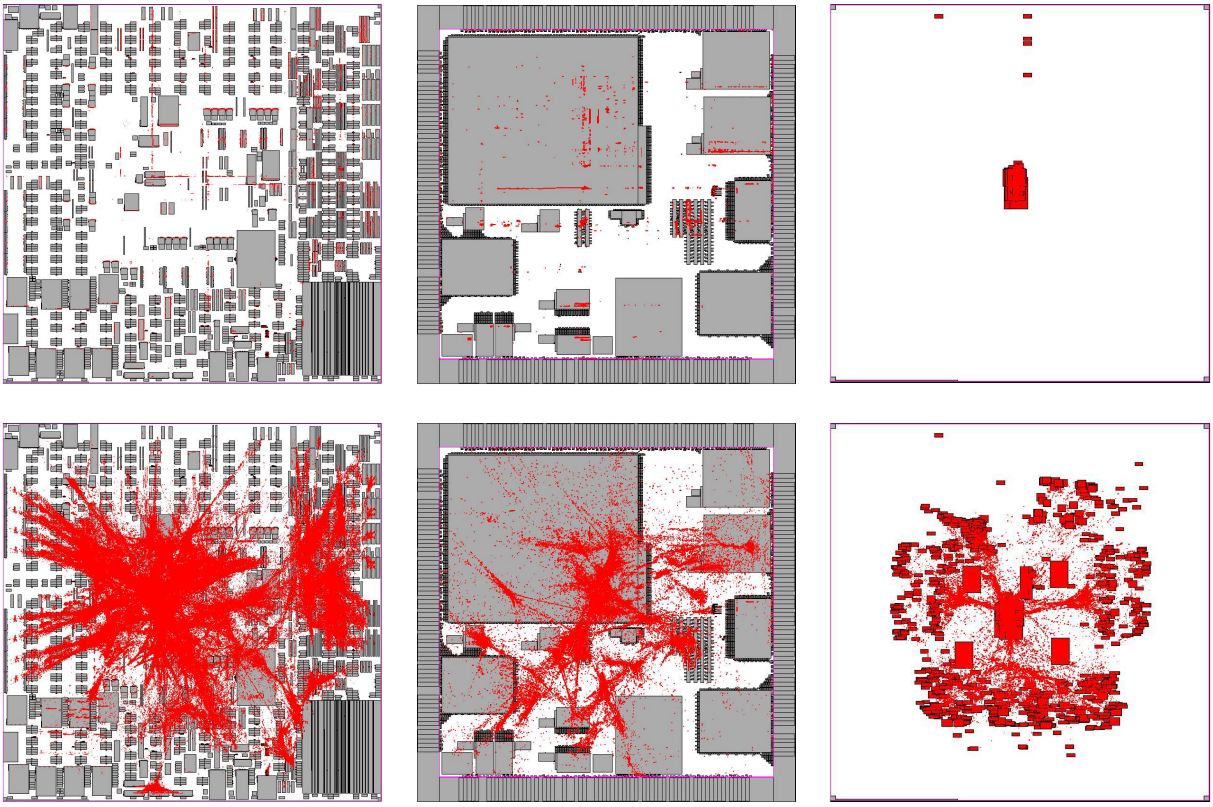

Figure 2: Placements minimizing linear netlength (upper pictures) and quadratic netlength (lower pictures).

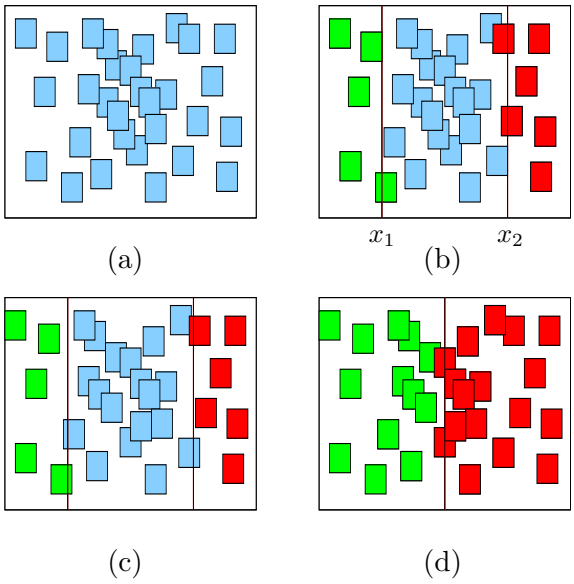

Figure 3: Iterative partitioning as described by Sigl, Doll and Johannes [1991].

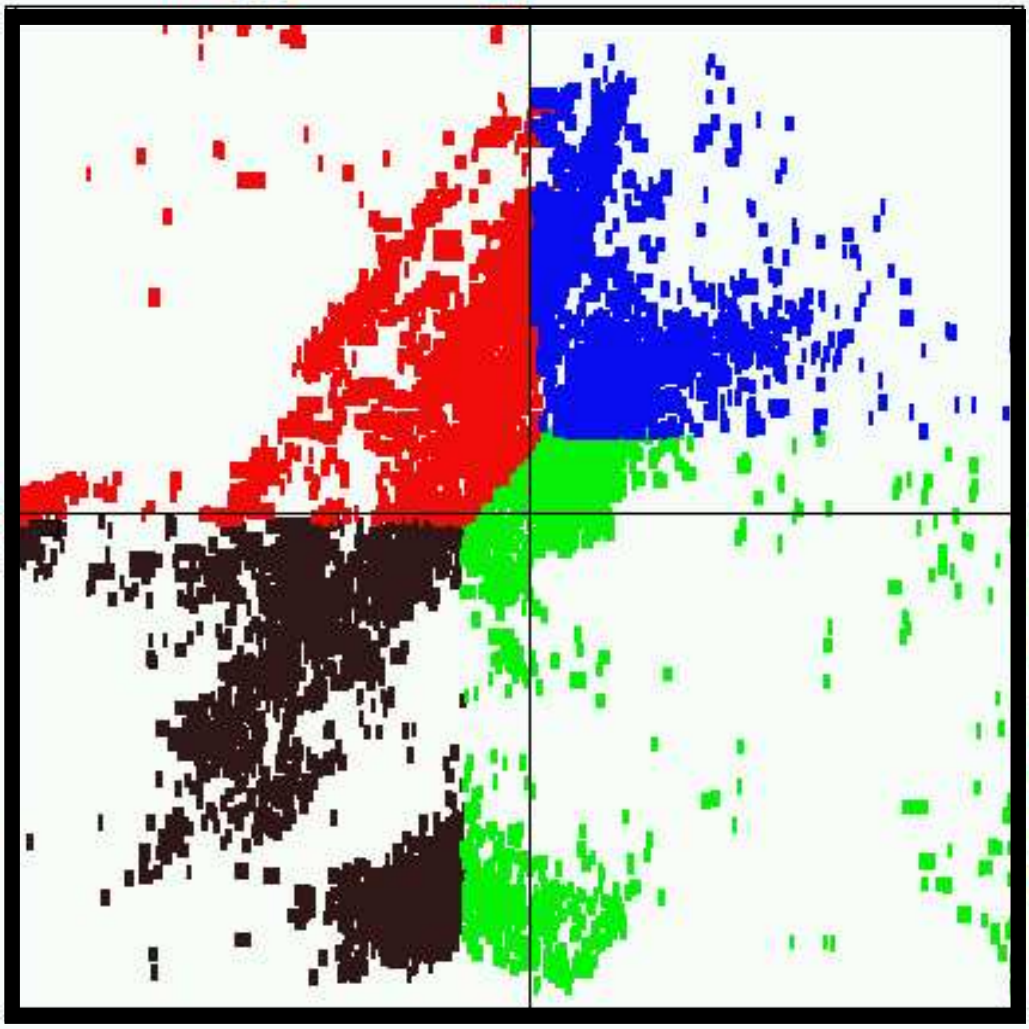

Figure 4: A set of cells partitioned by quadrisection (according to an American map).

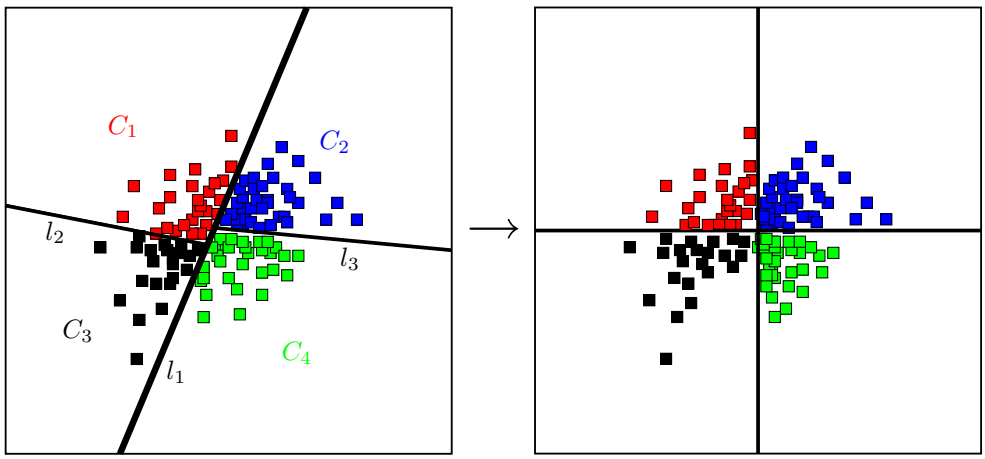

Figure 5: A "grid-warping" partitioning step.

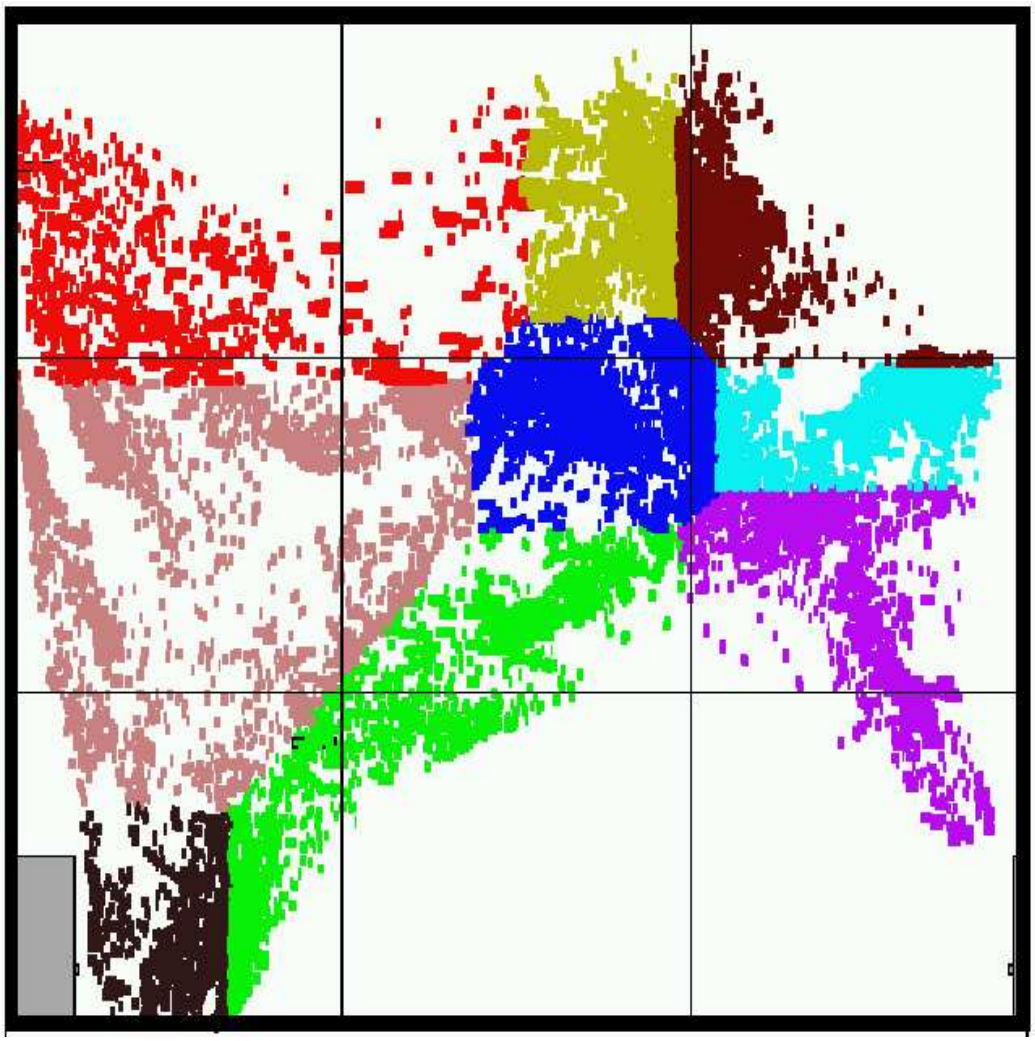

Figure 6: A set of cells partitioned by multisection.

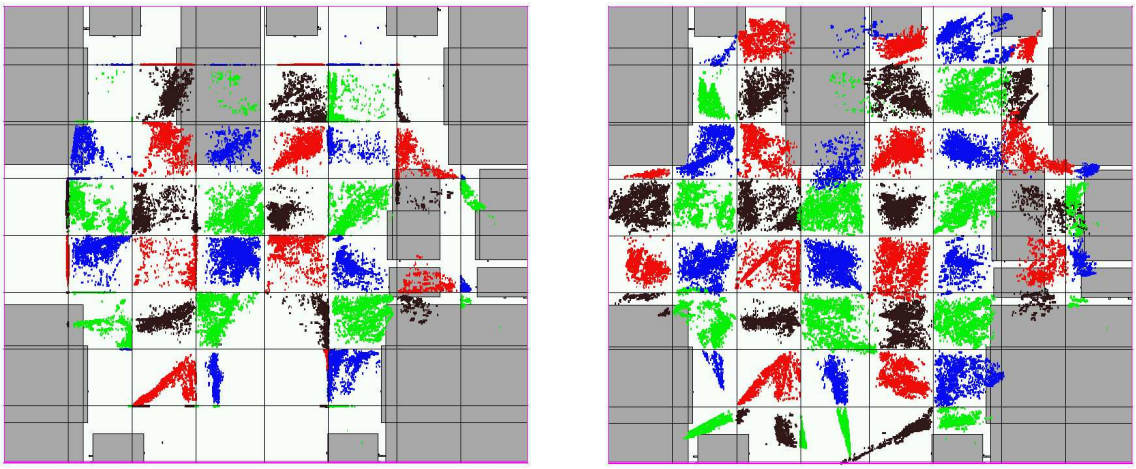

Figure 7: The effect of the constrained QP before the partitioning step.

Table 1:

	BB(V)	STEINER(V)	CLIQUE(V)	STAR(V)
BB(V)	1	1	1	1
STEINER(V)	$\frac{\lceil\sqrt{n-2}\rceil}{2}+\frac{3}{4}$	1	$\begin{cases}\frac{9}{8} & \text{for } n=4\\ 1 & \text{for } n\neq 4\end{cases}$	1
CLIQUE(V)	$\frac{\lceil\frac{n}{2}\rceil\lfloor\frac{n}{2}\rfloor}{n-1}$	$\frac{\lceil\frac{n}{2}\rceil\lfloor\frac{n}{2}\rfloor}{n-1}$	1	1
STAR(V)	$\lfloor\frac{n}{2}\rfloor$	$\lfloor\frac{n}{2}\rfloor$	$\frac{n-1}{\lceil\frac{n}{2}\rceil}$	1