Optimal Phase Conflict Removal for Layout of Dark Field Alternating Phase Shifting Masks*

Piotr Berman†, Andrew B. Kahng, Devendra Vidhani, Huijuan Wang and Alex Zelikovsky‡

UCLA Dept. of Computer Science, Los Angeles, CA 90095-1596, {abk, vidhani, huijuan } @cs.ucla.edu
†Dept. of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802-6106, berman@cse.psu.edu
‡Dept. of Computer Science, Georgia State University, University Plaza, Atlanta, GA 30303, alexz@cs.gsu.edu

Abstract

We describe new, efficient algorithms for layout modification and phase assignment for dark field alternating-type phase-shifting masks in the single-exposure regime. We make the following contributions. First, the problem of minimizing the number of phase conflicts that must be removed to ensure 2-colorability of the conflict graph is transformed to the T-join problem. Second, we give new exact and approximate algorithms for the resulting T-join problem. The exact algorithm is linear in space and substantially faster for sparse graphs than previously known algorithms. The output of the approximate algorithm is much closer to the optimum than the algorithm suggested in [9]. These algorithms are applied within a "one-shot" phase assignment method suggested in [9] for co-optimization of layout and phase assignment for alternating phase-shifting masks. Preliminary computational experience is promising.

1 Introduction

Phase-shifting mask (PSM) technology enables the clear regions of a mask to transmit light with prescribed phase shift. Consider two adjacent clear regions with small separation, and respective phase shifts of 0 and 180 degrees. Light diffracted into the nominally dark region between the clear regions will interfere destructively; the improved image contrast leads to better resolution and depth of focus. All PSM variants employ this basic concept, which was proposed by Levenson et al. [10] in 1982 (see Figure 1). Along with optical proximity correction, PSM is enabling to the subwavelength optical lithography upon which the next several process generations depend. Our work, like that of Moniwa et al. [12, 13] and Ooi et al. [15, 16], pertains to the *dark field, alternating* (or *Levenson-type*) phase-shifting mask technology with negative photoresist. In particular, we seek methods compatible with *single-exposure* alternating PSM.

As in previous work [15], we use the positive constants $b < B$ to define a simplified relationship between printability and the distance between two clear regions. The distance between any two features cannot be smaller than b without violating the minimum spacing design rule. If the distance between two features is at least b but smaller than B, the features are in *phase conflict*.[1] Phase conflict can be resolved by assigning opposite phases to the conflicting features. In other words, B defines the minimum spacing rule when two features have the same phase. If the distance between two features is greater than B, there is no phase conflict and the features can be assigned arbitrary phases. Note that the values of b and B are layer-dependent. We also let $w \geq b$ denote the minimum allowed width of any feature on the layer of interest. Finally, we assume that all features are rectilinearly oriented (all edges axis-parallel) polygons.

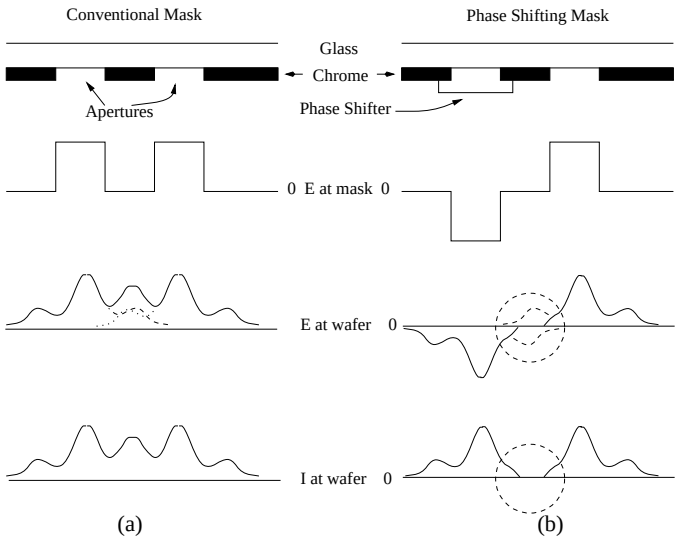

Figure 1: Comparison of diffraction optics of conventional and phase-shifting masks. E denotes electric field and I denotes intensity. With the conventional mask (a) light diffracted by two adjacent apertures constructively interferes, increasing the light intensity in the dark area of the wafer between the apertures. With the phase-shifting mask (b), the phase shifter reverses the sign of the electric field, and destructive interference minimizes light intensity at the wafer in the dark area between apertures.

The Phase Assignment Problem: Given a layout, assign phases to all features such that no two conflicting features are assigned the same phase.

2 Constructing and Planar Embedding of the Conflict Graph

We now show how to construct a *conflict graph* corresponding to a given layout, such that the assignment of phases to features corresponds to the coloring of vertices of the conflict graph. We also describe a planar embedding of the conflict graph. For a given layout of polygonal features, the *conflict graph* $G = (V, E)$ is constructed by defining a vertex for each feature, and introducing an edge between two vertices exactly when the corresponding features are in phase conflict. The conflict graph can be constructed in $O(n \log n)$ time, where n is the total number of segments in all polygon boundaries. We implement the following construction.

*This work was supported by a grant from Cadence Design Systems, Inc.

[1] More precisely, two features are in phase conflict if (i) there is no pair of points, one from each feature, whose separation is less than b; and (ii) there is some pair of points, one from each feature, whose separation is less than B.

- Slice each feature (polygon) into rectangles by vertical cuts through all polygon vertices, maintaining a pointer from each rectangle to its containing polygon.
- Bloat each rectangle by distance $B/2$.
- Using sweepline and interval trees, detect conflicts between polygons by finding overlapping pairs of rectangles that belong to different polygons.

Alternatively, one may detect intersections of *bounding boxes* of polygons, then check whether the corresponding polygons actually intersect.

In alternating PSM, we can remove all phase conflicts by assigning opposite phases to each pair of adjacent vertices in the conflict graph G. This is equivalent to 2-coloring the vertices of G with phase 0 and phase 180. For this to be possible, G must be bipartite, i.e., have no odd cycles. Hence, if the conflict graph G is not bipartite, our goal is to *delete* enough edges such that no odd cycles exist in the remaining modified conflict graph. Edge deletion in the conflict graph is achieved by *changing the placement of layout features* so that they no longer conflict. Thus, with alternating PSM technology we see that manufacturability creates highly non-obvious, non-local constraints on the layout. Efficient algorithms that we give below for removing odd cycles depend on *planarity* of G, an assumption that was justified in [9]. For the sake of completeness, we sketch the proof of the following theorem below.

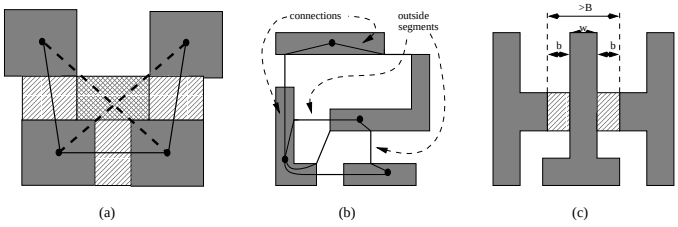

Figure 2: (a) We assume that there are no conflicts between diagonal pairs (dashed edges) in a set of four features if at least three other conflicts exist. (b) The subdivision of edges of the conflict graph. (c) There is no conflict between the left feature and the right feature because $B \leq 2b$.

Theorem 2.1 *Assume that (i) the minimum feature size w satisfies $w \geq b$, (ii) $2b \geq B$ and (iii) four rectangles which are pairwise (with the possible exception of one pair) in conflict do not have diagonal conflicts (see Figure 2(a)). Then, the conflict graph G is planar.*

Sketch of proof. For each feature, locate a representative node arbitrarily within the feature. For any two features in conflict, choose a pair of closest points on their boundaries and connect each of these points to the other and to the representative node of its own respective feature (see Figure 2(b)). In other words, each conflict edge is subdivided into two *connections* inside features and an *outside segment* between features. This subdivision does not affect planarity. For each feature, the connections between its representative node and all points on the boundary can be routed without intersections. Any outside segment is shorter than B, therefore, no outside segment can go into a feature and then leave it because it would then have length at least $2b \geq B$ (see Figure 2(c)). Thus if two conflict edges intersect, their outside segments intersect outside of features.

Suppose that (a,b) and (c,d) are two intersecting outside segments. Assume first that two points, say a and b, belong to the same feature. Because (a,b) and (c,d) intersect, $|a,b|+|c,d| > |a,d|+|b,c|$. Therefore, either (a,b) is longer than (a,d) or (c,d) is longer than (c,b), which contradicts the definition of the outside segment as a closest pair. If all four points belong to different features, then it can be shown that three pairwise distances other than (a,b) and (c,d) between these four points are smaller than B, which contradicts assumption (iii). ❑

Let K_4 and K_4' respectively denote an instance of four pairwise adjacent rectangles and an instance of four pairwise adjacent rectangles with one omitted boundary edge (see Figure 2(a)).Each intersecting pair of bloated rectangles should be connected by an edge, except when the edge corresponds to a diagonal of an instance of K_4 or K_4'. There are two reasons why we omit these diagonals. (1) Any valid phase assignment to vertices of the conflict graph is a valid phase assignment for the conflict graph without such diagonals, and vice versa.[2] (2) By Theorem 2.1, the conflict graph becomes planar after removal of diagonals in K_4 and K_4', and phases for planar graphs can be assigned efficiently in contrast to the case of arbitrary graphs (see Section 4).

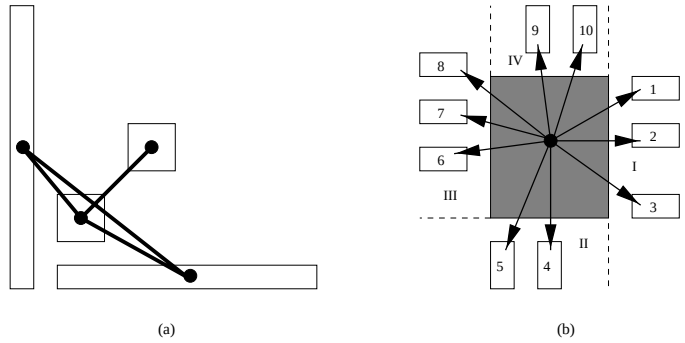

Figure 3: Obtaining a cyclic order on edges incident to a given vertex. (a) The order induced by the segments connecting the centers of adjacent rectangles is incompatible with a planar embedding. (b) The order which induces a correct planar embedding.

To fully exploit the planarity of the conflict graph once diagonals have been removed, the conflict graph should be *embedded* into the plane. Embedding is fully determined by the cyclic order of edges incident to each vertex (see [18]), i.e., by the cyclic order of vertices adjacent to a given vertex. Note that the order induced by segments connecting the centers of adjacent rectangles may be incompatible with a planar embedding, e.g., if rectangles have large aspect ratio (see Figure 3(a)). We use the following cyclic order of adjacent rectangles (see Figure 3(b)). All rectangles adjacent to a given rectangle R are partitioned into four groups consisting of: (i) rectangles positioned to the right of R (i.e., having left side to the right of the right side of R); (ii) rectangles positioned below R which do not belong to group (i); (iii) rectangles positioned to the left of R which do not belong to group (ii); and (iv) rectangles positioned above R which do not belong to groups (i) and (iii). In the ordering, rectangles from group (i) precede those from group (ii), which in turn precede those from group (iii), which in turn precede those from group (iv). Inside their respective groups, the rectangles are sorted (i) in decreasing order of their y-coordinates, (ii) in decreasing order of their x-coordinates, (iii) in increasing order of their y-coordinates, and (iv) in increasing order of their x-coordinates. After the correct planar embedding is established, all instances of K_4 and K_4' which cause edge intersections can be detected, and their edge intersections removed, in a straightforward way.

[2] In the conflict graph with diagonals removed from any instance of K_4 (respectively K_4'), any valid phase assignment will assign different phases to endpoints of each of the four (respectively three) edges left after removal of diagonals. Therefore, the two endpoints of each diagonal are assigned the same phase, and endpoints of different intersecting diagonals are assigned opposite phases. Our understanding is that for such configurations, this phase assignment yields acceptable feature resolution.

3 Removing Odd Cycles From the Conflict Graph

In this section we address the question of removing odd cycles from the conflict graph. We discuss several known methods suggested by Moniwa et al. and Ooi et al. as well as methods recently suggested in [9].

Moniwa et al. [12] and Ooi et al. [15] first posed the phase assignment problem and suggested methods of detecting cases when there is no valid phase assignment, i.e., when the conflict graph contains odd cycles. Subsequently, Moniwa et al. [13] and Ooi et al. [16] suggested two interactive methods which fully exploit information in the mask layout. The heuristic of Moniwa et al. [13] first constructs the conflict graph G, then creates a list of all odd cycles in G using an enumerative approach. The heuristic then iteratively finds and deletes the edge that is in the greatest number of minimum-length odd cycles. Deletion is accomplished by increasing the lower bound on separation between the corresponding features, and then applying a compactor to perturb the shape or position of these features. This enumerative approach may be feasible for the cell layout editing context, but likely does not scale to large instances since the number of odd cycles can be exponential in the size of the layout. Moreover, the heuristic does not necessarily delete the minimum number of edges, nor will it necessarily select edges whose deletion will have minimum impact on the layout.

Ooi et al. [16] also suggested a compaction-based method which (i) produces a symbolic layout from the mask layout; (ii) performs phase assignment in the symbolic layout; and (iii) compacts the symbolic layout using minimum spacing design rules consistent with the phase assignment. The advantage of the compaction-based method is that it is fully automated and guarantees to remove all odd cycles from the conflict graph. On the other hand, the phase assignment step is relatively oblivious to details of the mask layout; the ensuing compaction step may not minimize distortion of the original layout.

Recently, the following approaches to layout modification and phase assignment for alternating PSM were proposed in [9].

Iterative coloring and compaction. This approach generalizes the approaches of Ooi et al. [16] and Moniwa et al. [13]. Initially, the layout is constrained only by the minimum-spacing design rule, i.e., no two features can be less than distance b apart. Then the following three steps are iteratively applied until the conflict graph G becomes bipartite:

(i) compact the layout and find the conflict graph G;

(ii) find a (minimum) set of edges whose deletion makes the conflict graph G 2-colorable; and

(iii) add a new spacing constraint (to be enforced by the compactor) for each edge in this (minimum) set, such that the pair of features connected by this edge must be separated by distance at least B.

"One-shot" phase assignment. This is an improvement over the method in Ooi et al. [16]. It consists of the following steps:

(i) find the conflict graph G;

(ii) find a (minimum) set of edges whose deletion makes the conflict graph G 2-colorable;

(iii) assign phases such that only the conflict edges in this (minimum) set connect features of the same phase; and

(iv) compact the layout with "PSM design rules", i.e., minimum separation B between features that are assigned the same phase, and minimum separation b between features that are assigned different phases.

The Minimum Distortion Problem

In general, when an optimal phase assignment is found, each conflict edge that separates two same-phase features induces a minimum spacing requirement of B between the features. Such a requirement is passed to compaction in the form of a spacing constraint. To fully exploit compaction technology and achieve optimal algorithms for phase assignment with *minimum layout distortion*, we may assign different weights to different conflict edges. With the methods for optimal odd cycle removal that we develop below, even if the conflict edges have different weights it is still possible to find the minimum-weight set of conflict edges whose deletion makes the conflict graph bipartite. During compaction we may detect spacing constraints (between feature pairs) that are on *critical paths*, i.e., constraints that when increased will directly increase the size of the layout. It is reasonable to assign larger weight to conflict edges corresponding to such constraints; lesser weight should be assigned to conflict edges that do not lie on critical paths (see Figure 4). Finally, we observe that if several methods to delete edges and eliminate odd cycles are combined, the weight of a given conflict edge should reflect the minimum possible cost of breaking that edge using any of the available methods.

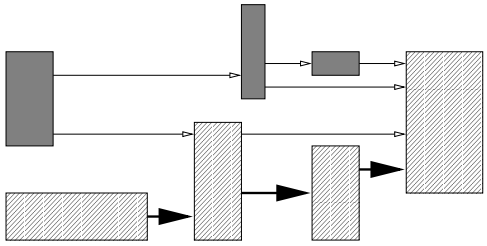

Figure 4: Critical path between leftmost and rightmost features consists of thick edges. Thin edges on non-critical paths may be broken for free.

Optimal phase assignment can be found by solving the following problem.

The Minimum Distortion Problem: Given a planar graph $G = (V, E)$ with weighted (multiple) edges, find the minimum-weight edge set M such that the graph $(V, E - M)$ contains no odd cycles.

After the Minimum Distortion Problem is solved, i.e., the set of edges M is determined and deleted, the valid assignment of phases can be found using breadth-first search. For each connected component of the conflict graph (with each edge weight set to 1), breadth-first search starting from arbitrary vertex v determines the distance from v to each other vertex u. If the distance from v to u is even, then u is assigned the same phase as v; otherwise, v is assigned opposite phase. Such breadth-first search can be performed in linear time.

4 Exact and Approximate Algorithms for the Minimum Distortion Problem

We now present exact and approximate algorithms for finding the minimum-cost set of conflict edges whose deletion eliminates all odd cycles, i.e., solving the Minimum Distortion Problem. Hadlock [5] and Orlova et al. [17] proposed the first exact algorithm, which we describe in Section 4.1. Unfortunately their method cannot be use in practice because it relies on finding all shortest paths between vertices of the dual graph of the conflict graph. This implies using quadratic memory as well as cubic runtime. In Section 4.2 we present a new fast exact algorithm with linear memory and subquadratic runtime. Section 4.3 discusses approximate algorithms for the Minimum Distortion Problem. The *Voronoi Algorithm* described in [9] is based on Voronoi

graphs and is very fast, but does not always find the optimal matching. The algorithm suggested by Ooi et al. [16] may be viewed as an even faster greedy approximation, but has very poor performance (see Section 5.1). Section 4.3 describes these two algorithms as well as a new *iterated Voronoi algorithm*.

4.1 Exact Algorithm

We use the following definitions. A *geometric dual* of an embedded planar graph $G =< V, E >$ is a multigraph $D =< F, E >$ in which nodes are the faces of G. If f, g are two faces of G, i.e. two nodes of D, then an edge $e \in E$ of G connects f with g if e belongs to the both faces f and g. A *reduced dual* of G is a graph $D =< F, E >$ obtained from D by deleting all but one of the edges that connect a given pair of nodes. The undeleted edge must be the one of minimal weight.

The Optimal T-join Problem [2]: Given a graph G with weighted edges, and a subset of nodes T, find a minimum weight edge set A such that a node u is incident to an odd number of edges of A iff $u \in T$.

Lemma 4.1 *The Minimum Distortion Problem for a planar graph G is equivalent to the Optimal T-join problem in the reduced dual graph of G.*

Proof. To eliminate all odd cycles it is sufficient to eliminate odd faces of the planar graph G (see Figure 5). The odd faces of G correspond to odd-degree vertices of D. Any edge elimination in G corresponds to edge contraction in D, i.e. two faces which share the deleted edge collapse into a single face. In particular, if we eliminate a set of edges A in G, then the resulting nodes of the modified $D' =< F', E \setminus A >$ will correspond to connected components of $< F, A >$, a subgraph of the original $D =< F, E >$ induced by edges from $A \subseteq E$. Given such a component with sum of node degrees d and k edges, the corresponding node has degree $d - 2k$ (each edge is counted twice in sum of all degrees). Thus A is a feasible solution if and only if each connected component of $< F, A >$ contains an even number of odd nodes (odd faces of G). Moreover, for each feasible solution $A \subseteq E$ there exists a feasible solution $A \subset E$ with weight that is not larger; we obtain A from A by replacing multiple edges connecting a pair of nodes/faces f and g with a single edge of minimum weight.

If we define T to be the set of odd faces of G, then finding the minimum cost feasible solution is the same as solving the Optimum T-join Problem for D. ❑

One can find a discussion of the T-join Problem in Cook et al. [2], pp. 166-181. Hadlock [5] and Orlova et al. [17] solved this problem using the following reduction.

Lemma 4.2 *The Optimal T-join problem for a graph with n nodes can be reduced to Minimum-Weight Perfect Matching problem in a complete graph with $|T|$ nodes.*

Proof. Every minimal T-join is the union of edge sets of edge disjoint paths that, viewed as edges connecting their endpoints, provide a perfect matching of set T (see [2], p. 168). Thus we can find an optimal T-join by finding the minimum weight perfect matching in the graph having node set T and edge weights being the shortest path lengths in the original graph. ❑

4.2 Speeding-Up the Exact Algorithm

The second reduction that defines the algorithm of Hadlock and Orlova et al. has two drawbacks. First, the reduction itself can be slow, because finding all pairwise distances between vertices of T is too time- and memory-consuming. Additionally, the resulting instance of the Minimum-Weight Perfect Matching Problem may have many more edges than necessary, and thus itself has too high complexity to be used in practice. We now provide an approach that is much more efficient in the case of sparse graphs (note that planar graphs are always sparse, because the number of edges is less than six times larger than the number of nodes).

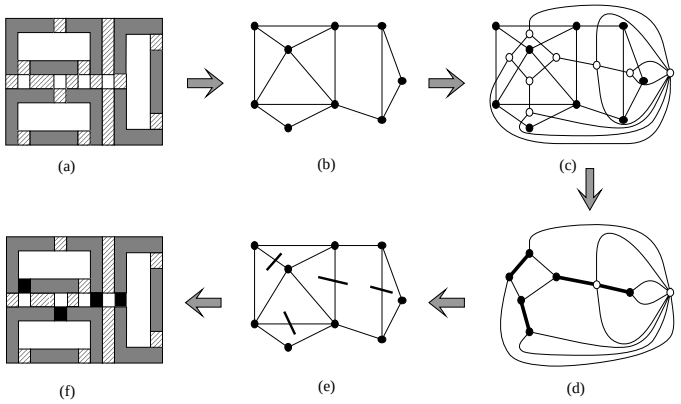

Figure 5: From the conflicts between features (a), the conflict graph is derived (b). The dual graph (c) is constructed. The vertices of odd degree are matched using paths in the dual graph (d), and the corresponding conflict edges are determined (e). Finally, the minimum set of conflicts to be deleted is determined (f).

Lemma 4.3 *The Optimal T-join problem for a graph with n nodes (and T with t nodes) and m edges can be reduced to the Minimum-Weight Perfect Matching problem in a graph with $4m - 2n - t$ nodes and $7m - 5n - t$ edges.*

Proof. We can eliminate from consideration nodes of degree 0 and 1. Other nodes will be replaced with *gadgets*. Suppose a node v has neighbors $u_1, \ldots, u_d$. Then the gadget Γ_v that replaces the original node v will contain nodes $(v,u_1), \ldots, (v,u_k)$ and some other nodes. An original edge $\{u,v\}$ will be replaced with $\{(u,v),(v,u)\}$, with the same cost. We will also add a number of 0-cost edges to connect each gadget. This translation will be valid because it will satisfy: (i) if M is a perfect matching in the new graph, then the set of all original edges with replacements in M forms a T-join; and (ii) if A is a T-join in the original graph, then the set of replacements of the edges of A, together with some 0-cost edges, forms a perfect matching.

Property (i) will be assured in the following manner: Γ_v will have an odd number of nodes if and only if $v \in T$. Thus if $v \in T$, each perfect matching must contain an odd number of edges with exactly one endpoint in Γ_v, i.e., an odd number of replacements of edges that are incident to v. Similarly, if $v \notin T$, the number of such edges must be even.

Property (ii) will be assured as follows. Consider a gadget Γ_v. We will say that the nodes of the form (v,u) are the *contacts* of Γ_v. After removing any number of contacts from Γ_v, the set of remaining nodes of Γ_v will contain a Hamiltonian path, and thus it will contain a perfect matching (of 0-cost) if its size is even.

One can easily see in Figure 6 that our gadgets indeed satisfy the claimed properties. The above argument does not deal with original nodes of degree 2; this case is simple and follows from a slightly different argument. ❑

To solve the problem resulting from the application of Lemma 4.3, we can apply the best algorithm known to date, by Gabow and Tarjan [4]

Theorem 4.4 *There exists an algorithm that solves the Minimum Distortion Problem in time $O((n\log n)^{3/2})\alpha(n)$, where α is the inverse of*

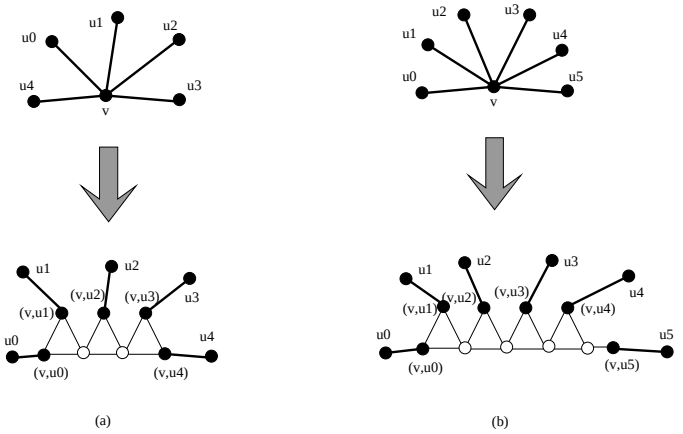

Figure 6: Replacing a vertex v of the dual graph with a gadget. The original (thick) edges have weight 1 and the auxiliary (thin) edges have weight 0. The contacts are represented by black circles. (a) If $v \in T$, then it is replaced with a chain of triangles. (b) If $v \notin T$ then there is an edge attached to the last triangle.

Ackerman function.

The oldest polynomial time algorithm for Minimum-Weight Perfect Matching, invented by Edmonds, establishes a running time of $O(n^4)$. Nevertheless, in practice, the latter algorithm behaves much better than the proven worst case and is the basis for almost all implementations that solve the Perfect Matching problem. In our code, we integrate the implementation of Cook and Rohe [3].

4.3 Fast Approximation Algorithms

The exact algorithm from the previous subsection may still be too slow for processing very large layouts. A naive method, used by Ooi et al. [16], is to color the conflict graph in a breadth-first manner. In other words, assign the first color to an arbitrarily chosen feature from each connected component of the conflict graph. Then, using breadth-first search, visit all features and assign opposite color to all features adjacent to already visited features. In case of a non-bipartite conflict graph, some conflict edges will have the same color assigned to different endpoints. The number of such edges equals the number of edges that should be deleted in order to make the conflict graph bipartite. This approach, which we call the *Greedy Algorithm*, is very fast but can give very large error, i.e., the number of conflict edges with the same phase assigned to both ends can be several times larger than for the exact algorithm (see Section 5.1).

In [9] a fast algorithm based on the Voronoi graph paradigm of [11] was suggested. This *Voronoi Algorithm* is much faster than the exact algorithm but may leave some odd cycles.

Voronoi Algorithm:

1. Build a graph G' which is the geometric dual of G. Each vertex in G' corresponds to a face in G. Each edge e' between two vertices of G' corresponds to the edge e that separates the two corresponding faces in G. The weight of an edge e' in G' is equal to the weight of the corresponding edge e in G.

2. Let T to denote this set of vertices in G' corresponding to odd faces of the graph G. Partition all vertices in G' into the *Voronoi regions* of the vertices in T [11] such that:

   (i) each Voronoi region contains exactly one vertex from T, which is the *center* of the Voronoi region, and

   (ii) each even-degree vertex belongs to the Voronoi region that contains the closest vertex in T (break ties arbitrarily).

3. Construct the *Voronoi graph* $Vor(G')$ in which each vertex represents a Voronoi region, and two vertices R_1 and R_2 are adjacent if a shortest path in G' between the corresponding region centers is completely contained in the two regions.

4. Find a minimum-weight maximum matching M in $Vor(G')$.

5. Find the edge set VM in G that is the union of paths in G' corresponding to edges of M, which in turn correspond to the edges of the minimum weighted matching in $Vor(G')$.

6. Delete edges VM from the conflict graph G and run the Greedy Algorithm.

In this work, we suggest an improved *Iterated Voronoi Algorithm* which is still fast, but which deletes a smaller number of edges from the conflict graph. Instead of running the Greedy Algorithm at the final step of the Voronoi Algorithm, the Iterated Voronoi Algorithm deletes the edges of the dual graph G' that correspond to the edges from VM, then iteratively finds the new Voronoi graph for vertices corresponding to the new set T. This iterative application of the Voronoi Algorithm is continued until T is empty, i.e., until the critical graph is bipartite. Our experimental results in Section 5 show that the Iterated Voronoi Algorithm remains sufficiently fast, and deletes fewer conflict edges, in comparison to the Voronoi Algorithm of [9].

5 Implementation Experience

We have implemented the "one-shot" method from Section 3;other methods including several proposed in [9] are also currently under investigation.

5.1 Software Implementation

Our system has been implemented in C++ on the Solaris 2.6, Sun CC 4.2 platform. Input is (hierarchical) GDSII that is converted to CIF, then read into an internal polygon database. There are two major software elements.

The *phase generator* finds a minimum-cost set of conflict edges for deletion, induces a phase assignment, and generates compaction constraints such that the layout remains consistent with the phase assignment. The code *Blossom-IV* for the Minimum-Weight Perfect Matching, by W. Cook and A. Rohe (1998) [3], was obtained from http://www.or.uni-bonn.de/home/rohe/matching.html.

In the one-shot flow, the phase generator creates compaction constraints as follows: if two features are assigned different phases, they have minimum separation b; otherwise, they have minimum separation B. (By contrast, in the iterative flow, constraints are specified only for pairs of features corresponding to broken conflict edges.)

The *graph-based compactor* (i) generates constraints between feature edges according to standard design rules and an efficient sweepline approach, (ii) adds constraints produced by the phase generator, and (iii) stores all constraints as edges of a constraint graph. The compactor then applies the Bellman-Ford algorithm to solve the constraint graph, i.e., obtain optimal x-coordinates of all edges of all features. Our implementation generally follows the description of leaf-cell compaction given by Bamji and Varadarajan [1]. We consider three types of constraints: (i) *shape* constraints fix the shape of features which cannot be changed according to design rules; (ii) *overlap* constraints ensure that features will remain electrically connected after compaction, and/or properly aligned between different layers; and (iii) *spacing* constraints enforce separation design rules (including PSM-specific separation rules between features of the same or of the different phase).

Note that we perform y-compaction the same way that we do x-compaction (after temporarily swapping x and y coordinates). In the

Testcases	Layout1		Layout2		Layout3	
#polygons/#edges	3769	12442	9775	26520	18249	51402

Algorithms	#conflicts	runtime	#conflicts	runtime	#conflicts	runtime
Greedy	2654	.56	2690	3.66	6168	5.38
Voronoi	2340	2.20	2064	4.69	5050	11.07
Iterated Voronoi	1828	2.35	1554	5.46	3494	13.51
Exact	1468	19.88	1346	16.67	2958	74.33

Table 1: Computational results for phase assignment of layouts with various sizes. The top row gives the number of polygons and the number of conflict edges for each testcase. The bottom four rows contain the numbers of unresolved conflict edges (i.e., the numbers of pairs of polygons within distance B with the same phase, which must be resolved by perturbing the layout with a compactor) and runtimes for four phase assignment algorithms. All runtimes are in seconds for a 300 MHz Sun Ultra-10 workstation with 128MB RAM.

one-shot flow, we output the new positions of all features after compacting in x and y directions. (By contrast, in the iterative flow, we return control to the phase generator and continue the iteration until we obtain a valid phase assignment for all features.)

5.2 Computational Experience

Table 1 summarizes our computational experience with three layouts of different sizes and densities. All layouts were derived from industry standard-cell layouts. All runtimes are CPU seconds on a 300 MHz Sun Ultra-10 workstation with 128MB RAM. We see that our code can handle very large flat designs in reasonable time, and is a promising basis for phase assignment in alternating PSM at the block level, if not the full-chip level. Table 1 also confirms that the four algorithms studied exhibit a clear trade-off between runtime and the number of unresolved conflicts in the resulting valid phase assignment. Moreover, our new exact method significantly improves over the previous methods of [16] [9]: it reduces by 40% the number of unresolved conflicts, which correspondingly reduces the amount of layout modification needed in compaction.

In conclusion, we have suggested new optimal and approximate efficient algorithms for minimum-cost layout perturbation and conflict elimination in the dark field (negative photoresist, single exposure) alternating PSM context. Our approach has been integrated with a GDSII reader, polygon database and layout compactor. Our preliminary computational tests show that our code can assign phases to comparatively large designs in reasonable time. Other, potentially more powerful approaches to layout modification and phase assignment for alternating PSM are currently under investigation. We are also pursuing unified solutions to the dark-field and bright-field alternating PSM contexts.

Acknowledgments

We thank Andrew Caldwell, Stefanus Mantik and Igor Markov for critical assistance in our software development.

REFERENCES

[1] C. Bamji and R. Varadarajan, *Leaf Cell and Hierarchical Compaction Techniques*, Kluwer Academic Publishers, Boston, 1997.

[2] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank and A. Shrijver, *Combinatorial Optimization*, Willey Inter-Science, New York, 1998.

[3] W. Cook and A. Rohe, "Computing Minimum-Weight Perfect Matchings", manuscript, August, 1998. *http://www.or.uni-bonn.de/home/rohe/matching.html*.

[4] H. N. Gabow and R. E. Tarjan, "Faster scaling algorithms for general graph matching problems", Journal of the ACM 38 (1991) 815-853.

[5] F. O. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time", *SIAM J. Computing* 4(3) (1975), pp. 221-225.

[6] A. B. Kahng, G. Robins, A. Singh, H. Wang and A. Zelikovsky, "Filling and Slotting: Analysis and Algorithms", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, 1998, pp. 95-102.

[7] A. B. Kahng, G. Robins, A. Singh and A. Zelikovsky, "Filling Algorithms and Analyses for Layout Density Control", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, to appear.

[8] A. B. Kahng and H. Wang, "Toward Lithography-Aware Layout: Preliminary Litho Notes", manuscript, July 1997.

[9] A. B. Kahng, H. Wang and A. Zelikovsky, "Automated Layout and Phase Assignment Techniques for Dark Field Alternating PSM", *SPIE 11th Annual BACUS Symposium on Photomask Technology*, SPIE 1604 (1998), pp. 222-231.

[10] M. D. Levenson, N. S. Viswanathan and R. A. Simpson, "Improving Resolution in Photolithography with a Phase-Shifting Mask", *IEEE Trans. on Electron Devices* ED-29(11) (1982), pp. 1828-1836.

[11] K. Mehlhorn, "A Faster Approximation Algorithm for the Steiner Problem in Graphs", *Information Processing Letters* 27 (1988), pp. 125-128.

[12] A. Moniwa, T. Terasawa, N. Hasegawa and S. Okazaki, "Algorithm for Phase-Shift Mask Design with Priority on Shifter Placement", *Jpn. J. Appl. Phys.* 32 (1993), pp. 5874-5879.

[13] A. Moniwa, T. Terasawa, K. Nakajo, J. Sakemi and S. Okazaki, "Heuristic Method for Phase-Conflict Minimization in Automatic Phase-Shift Mask Design", *Jpn. J. Appl. Phys.* 34 (1995), pp. 6584-6589.

[14] J. Nistler, G. Hughes, A. Muray and J. Wiley, "Issues Associated with the Commercialization of Phase Shift Masks", *SPIE 11th Annual BACUS Symposium on Photomask Technology*, SPIE 1604 (1991), pp. 236-264.

[15] K. Ooi, S. Hara and K. Koyama, "Computer Aided Design Software for Designing Phase-Shifting Masks", *Jpn. J. Appl. Phys.* 32 (1993), pp. 5887-5891.

[16] K. Ooi, K. Koyama and M. Kiryu, "Method of Designing Phase-Shifting Masks Utilizing a Compactor", *Jpn. J. Appl. Phys.* 33 (1994), pp. 6774-6778.

[17] G. I. Orlova and Y. G. Dorfman, "Finding the Maximum Cut in a Graph", *Engr. Cybernetics* 10 (1972), pp. 502-506.

[18] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.