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

Piotr Berman, Andrew B. Kahng, Devendra Vidhani, Huijuan Wang, and Alexander Zelikovsky

***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, we suggest new two-coloring and compaction approach that simultaneously optimizes layout and phase assignment which is based on planar embedding of an associated conflict graph. We also describe additional approaches to cooptimization of layout and phase assignment for alternating PSM. Second, we give optimal and fast algorithms to minimize the number of phase conflicts that must be removed to ensure two colorability of the conflict graph. We reduce this problem to the T-join problem which asks for a minimum weight edge set A such that a node u is incident to an odd number of edges of A if u belongs to a given node subset T of a weighted graph. Third, we suggest several practical algorithms for the T-join problem. In sparse graphs, our algorithms are faster than previously known methods. Computational experience with industrial VLSI layout benchmarks shows the advantages of the new algorithms.**

***Index Terms*—Compaction, layout_verification, physical_design, VLSI.**

I. 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°. 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.* [13] in 1982 (see Fig. 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.* [15], [16], and Ooi *et al.* [18], [19] pertains to the *dark field*, *alternating-* (or *Levenson*-type) phase shifting mask technology with negative photoresist. It also matches very well with the requirements (in the positive photoresist technology regime) of damascene metallization, e.g., inlaid copper local interconnects. In particular, we seek methods compatible with *single exposure* alternating PSM.

As in previous work [18], 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 or equal to 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.

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

In this paper, we develop solutions to the problem of creating a phase assignable layout. As described in Sections II and III, this fundamentally corresponds to ensuring that there are no odd cycles in a so called *phase conflict graph* that is derived from the layout. When there are odd cycles, the layout must be perturbed so as to "break" all odd cycles; this is the problem of optimal *phase conflict resolution*. Our paper focuses on minimum cost breaking of all odd cycles in the conflict graph; this is the key problem in PSM layout design.

Today, PSM is applied only to "critical poly," i.e., transistor gates, to achieve smaller channel lengths and improved control of critical dimensions. For the "full chip" PSM that appears likely in the 150-nm generation and beyond, our new optimal algorithms for phase conflict resolution are readily applicable. For bright field PSM—i.e., the poly layer—our T-join based solution in Section IV applies with small modifications that are beyond the scope of our discussion. For dark field PSM—i.e., damascene local metal layers—our T-join based solution directly applies. For simplicity and clarity, our discussion will concentrate on the dark field context.

The rest of the paper is organized as follows. In Section II, we construct a conflict graph in which the phase assignment problem is reduced to node bicoloring. Section III discusses previously known and new methods of design modification which ensure bicolorability of the associated conflict graph.

Manuscript received June 1, 1999; revised November 11, 1999. This work was supported in part by a gift from Cadence Design Systems, Inc.. The work of P. Berman was supported by the National Science Foundation (NSF) under Grant CCR-9700053. The work of A. Zelikovsky was supported by Georgia State University (GSU) under Research Initiation Grant 00-013. This paper was recommended by Associate Editor D. Hill.

P. Berman is with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802-6106.

A. B. Kahng, D. Vidhani, and H. Wang are with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90095-1596.

A. Zelikovsky is with the Department of Computer Science, Georgia State University, University Plaza, Atlanta, GA 30303.

Publisher Item Identifier S 0278-0070(00)01793-0.

[1]More precisely, two features are in phase conflict if: 1) there is no pair of points, one from each feature, whose separation is less than b and 2) there is some pair of points, one from each feature, whose separation is less than B.

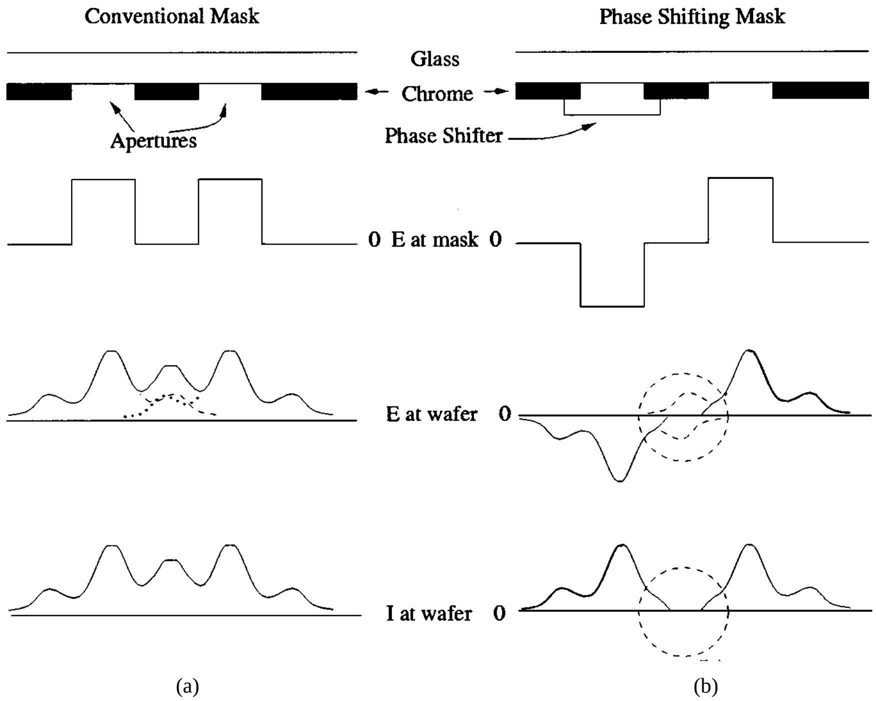


Fig. 1. Comparison of diffraction optics of conventional and phase shifting masks. E denotes electric field and I denotes intensity. With the conventional mask: (a) light diffracted by two adjacent apertures constructively interferes, increasing the light intensity in the dark area of the wafer between 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.

In Section IV, we reduce the phase assignment problem to the minimum perturbation problem in the conflict graph and to the T-join problem on the dual graph of the conflict graph. We also argue that the existing methods of solving the T-join problem are impractical. In Section V, we present new fast and practical algorithms for solving the T-join problem. These algorithms are based on reduction of the T-join problem to the minimum weight perfect matching problem via gadgets. Section VI is devoted to several previously known and new approximation methods for solving the T-join problem. Finally, in Section VII we discuss results of implementation of the previously known and new methods of phase assignment.

II. Construction 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 nodes 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 node for each feature, and introducing an edge between two nodes 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.

- Slice each feature (polygon) into rectangles by vertical cuts through all polygon nodes,[2] maintaining a pointer from each rectangle to its containing polygon.
- Bloat each rectangle by distance $B/2$.
- Using sweepline and interval tree, 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 nodes in the conflict graph G. This is equivalent to two coloring the nodes 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 nonobvious, nonlocal 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 [12]. In general, the conflict graph computed as described above may contain nonplanar local configurations like the one in Fig. 2(a). Using the process simulation tool *Nanosurfer* available from Numerical Technologies, Inc, we have confirmed that diagonal conflicts effectively do not exist because of interference effects. Therefore, to our understanding the deletion of intersecting diagonals from the conflict graph [using the $O(n)$

[2]Although the slicing is not strictly necessary, it greatly simplifies the construction as well as the planar embedding of the conflict graph. In fact, if slicing were not be performed in advance, the conflict graph construction and sorting of neighbors would implicitly include some procedure equivalent (at least in runtime) to slicing.

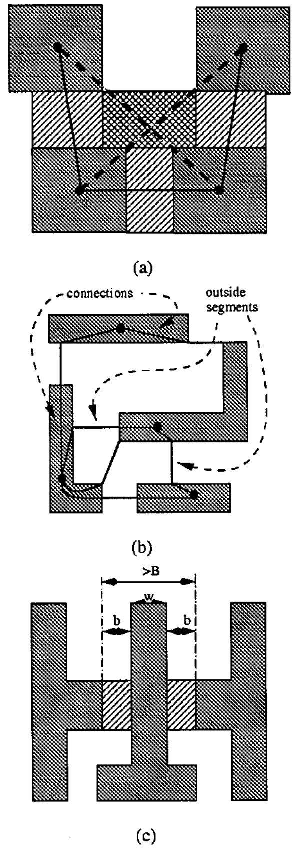


Fig. 2. (a) We assume that there are no conflicts between diagonal pairs (dashed edges) in a set of four features if at least three other conflicts exist, (b) the subdivision of edges of the conflict graph, and (c) there is no conflict between the left feature and the right feature because $B \leq 2b$.

procedure described below] will not compromise the effectiveness of the resulting PSM.

Theorem 2.1: Assume that: 1) $2b \geq B$ and 2) four rectangles which are pairwise (with the possible exception of one pair) in conflict do not have diagonal conflicts [see Fig. 2(a)]. Then, the conflict graph G is planar.

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 Fig. 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 Fig. 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 2). $\square$

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 Fig. 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 nodes of the conflict graph is a valid phase assignment for the conflict graph without such diagonals, and vice versa.[3], and 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.

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 node (see [21]), i.e., by the cyclic order of nodes adjacent to a given node. 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 Fig. 3(a)]. We use the following cyclic order of adjacent rectangles [see Fig. 3(b)]. All rectangles adjacent to a given rectangle R are partitioned into four groups consisting of: 1) rectangles positioned to the right of R (i.e., having left side to the right of the right side of R); 2) rectangles positioned below R which do not belong to group 1); 3) rectangles positioned to the left of R which do not belong to group 2); and 4) rectangles positioned above R which do not belong to groups 1) and 3). In the ordering, rectangles from group 1) precede those from group 2), which in turn precede those from group 3), which in turn precede those from group 4). Inside their respective groups, the rectangles are sorted 1) in decreasing order of their y-coordinates, 2) in decreasing order of their x-coordinates, 3) in increasing order of their y-coordinates, and 4) 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.

III. Approaches to Removing Odd Cycles from the Conflict Graph

Moniwa *et al.* [15] and Ooi *et al.* [18] 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, Ooi *et al.* [19] and Moniwa *et al.* [16] suggested interactive methods which fully exploit information in the mask layout. The heuristic of Moniwa *et al.* [16] first constructs the conflict graph G, then

[3]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.

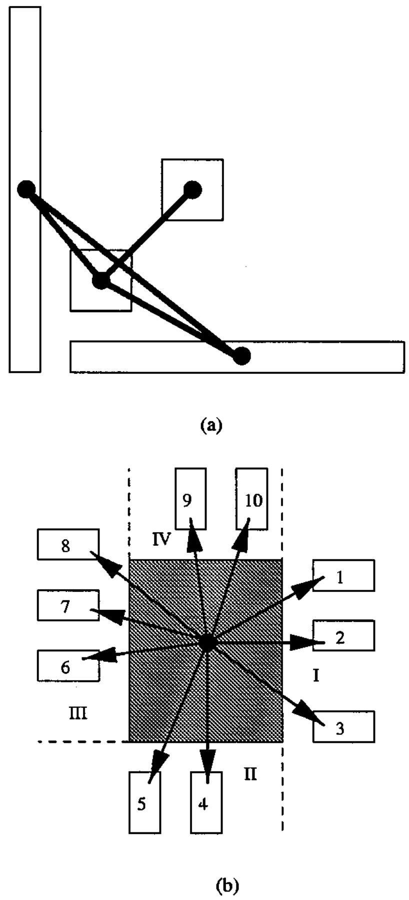


Fig. 3. Obtaining a cyclic order on edges incident to a given node: (a) The order induced by the segments connecting the centers of adjacent rectangles is incompatible with a planar embedding and (b) the order which induces a correct planar embedding.

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 (see Fig. 4). This 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.* [19] also suggested a compaction based method which: 1) produces a symbolic layout from the mask layout; 2) performs phase assignment in the symbolic layout; and 3) 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.

In the following, we focus on the following "one shot" approach (Fig. 5), which improves over the method in Ooi *et al.* [19]. (Other techniques, including several of a speculative nature, have been proposed in [12]; our present work does not address these.) Initially, we constrain the layout only by the minimum spacing design rule, i.e., no two features can be less than distance b apart. We then: 1) find the conflict graph G; 2) find the minimum set of edges whose deletion makes the conflict graph G two colorable; 3) assign phases such that only the conflict edges in the minimum set connect features of the same phase; and 4) 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.

IV. The Minimum Perturbation and T-Join Problems

The overall objective of PSM layout design is to achieve minimum area layout while maintaining PSM feasibility. Our "one shot" flow assumes that an automated custom or migration flow will essentially start out by being "optimistic," i.e., by assuming that feature widths and spacings can be scaled down to values achievable using PSM. The optimistic assumption will typically result in a design that cannot be phase assigned, due to the presence of odd cycles. This induces a "minimum perturbation" problem: we seek a "minimum perturbation" of the layout, in terms of odd cycle breaking and resultant B spacing constraints between pairs of features that previously had b spacings.[4]

Recall that when a 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 perturbation, we must take into account that different changes may have different impact on the resulting layout area.

The methods for optimal odd cycle removal that we develop below will find the minimum weight set of conflict edges whose deletion makes the conflict graph bipartite, even if the conflict edges have different weights. Thus, as we minimize the number of layout changes (i.e., new spacing requirements in compaction), we believe that it will be helpful to assign larger weights to those conflict edges whose resolution will cause larger increase in the layout area. A specific recipe would detect spacing constraints (between feature pairs) that are on *critical paths* in the compaction, i.e., constraints that when increased will directly increase the size of the layout. It would be 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 Fig. 6). Finally, 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.[5]

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

[4]An underlying—and natural—assumption is that adding fewer such "PSM constraints" will lead to less overall layout area (i.e., after compaction).

[5]We have not implemented this particular connection between our compactor and the conflict graph definition. Our discussion is simply to motivate the following formulation of the minimum perturbation problem.

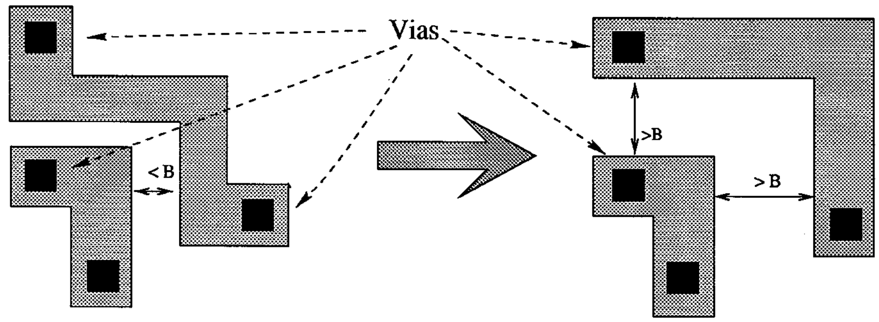


Fig. 4. Changing a shape without changing positions of vias.

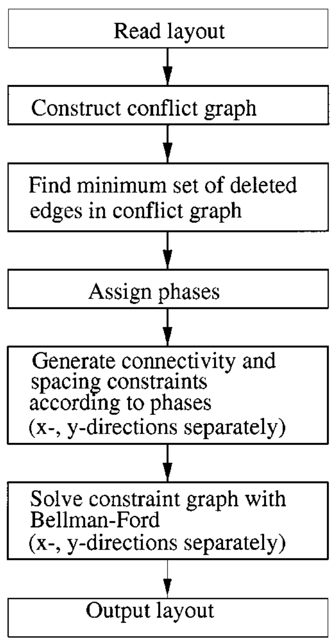


Fig. 5. Flow for the "one shot" method. Note that the one shot approach applies compaction separately in the x- and y-directions to enforce the given phase assignment.

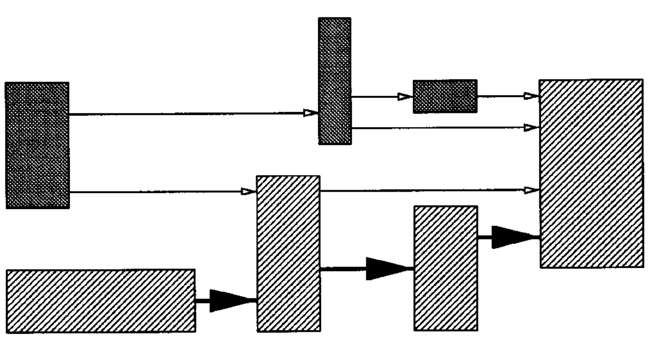


Fig. 6. Critical path between leftmost and rightmost features consists of thick edges. Thin edges on non critical paths may be broken for free.

1) The Minimum Perturbation 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 perturbation 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 (the weight of each edge is set to one), starting from arbitrary node v breadth first search determines the distance from v to each other node u. If the distance from v to u is even, then u is assigned the same phase as v; otherwise, u is assigned the opposite phase. Such breadth first search can be performed in linear time.

The minimum perturbation problem is closely related to the well known T-join problem [4].

2) The T-Join Problem [4]: Given a graph G with weighted edges, and a subset of nodes T, $|T|$ is even, find a minimum weight edge set A such that a node u is incident to an odd number of edges of A if $u \in T$.

The minimum perturbation problem can be reduced to the T-join problem in the following way. We use the following definitions. A *geometric dual* of an embedded planar graph $G = \langle V, E\rangle$ is a multigraph $D = \langle F, E\rangle$ in which nodes are the faces of G. If f, g are two faces of G, i.e., two nodes of D, than an edge of G connects f with g if it belongs to both of them. A *reduced dual* of G is a graph $\overline{D} = \langle F, \overline{E}\rangle$ 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.

Lemma 4.1: The minimum perturbation problem for a planar graph G is equivalent to the 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 Fig. 7). The odd faces of G form odd degree nodes of D. Any edge elimination in G corresponds to edge contraction in D. In particular, if we eliminate a set of edges A in G, then the resulting nodes of (modified) D will correspond to connected components of $\langle F, A\rangle$. Given such a component with sum of node degrees d and k edges, the corresponding node has degree $d - 2k$. Thus, A is a feasible solution if each connected component of $\langle F, A\rangle$ contains an even number of odd nodes (odd faces of G). Moreover, for each feasible solution $A \subset E$ there exists a feasible solution $\overline{A} \subset \overline{E}$ with weight that is not larger; we obtain $\overline{A}$ 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 T-join Problem for $\overline{D}$. □

The T-join problem was reduced by Hadlock [8] and Orlova and Dorfman [20] to the minimum weight perfect matching problem.

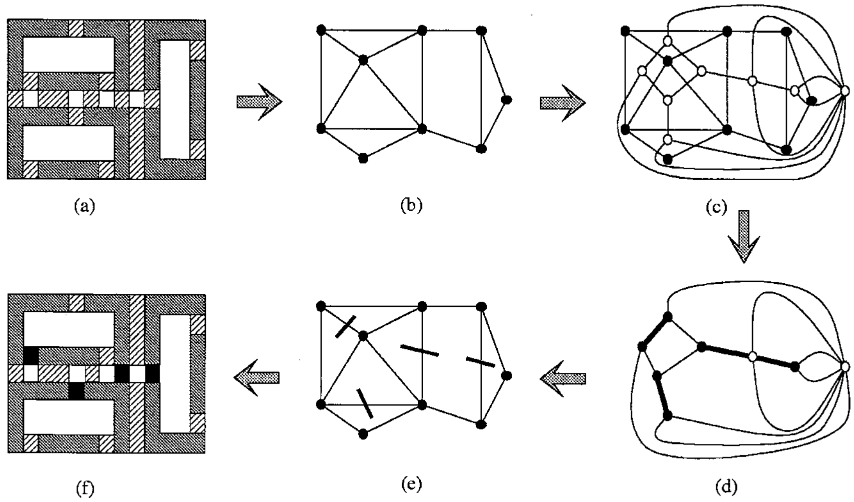


Fig. 7. From (a) the conflicts between features, (b) the conflict graph is derived. The dual graph (c) is constructed. (d) The nodes of odd degree are matched using T-join in the dual graph, and (e) the corresponding conflict edges are determined. Finally, (f) the minimum set of conflicts to be deleted is determined.

Lemma 4.2: The 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 [4, p. 168]). Thus, every minimal T-join corresponds to a perfect matching, with the same cost, in a complete graph with node set T and edge weights defined as the shortest path lengths in the original graph. Conversely, every perfect matching in the new graph yields a T-join considering the paths that correspond to its edges, and taking the edges of the original graph that belong to an odd number of these paths, obviously the cost of this T-join is not larger than the cost of the matching. Consequently, the minimum cost perfect matching must correspond to a minimum cost T-join. □

The reduction defined in Lemma 4.2 has two drawbacks. First, the reduction itself can be slow, because finding all pairwise distances between nodes of T is too time and memory consuming. Additionally, the resulting instance of minimum weight perfect matching problem may have many more edges than necessary and, thus, itself is too difficult to be used in practice. The remainder of this work provides several approaches that are 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).

V. Fast Optimal Algorithms for the T-Join Problem

In this section we present new reductions of the T-join problem to the minimum weight perfect matching problem, which yield a faster exact algorithm for the T-join problem in sparse graphs.

We start from simple reductions that allow to reduce the problem size in many practical instances, moreover, they eliminate special cases that would complicate the description of *gadgets* that are used by the subsequent reduction. Next, we will show a gadget based reduction that uses very simple gadgets and has simple correctness proof. This reduction yields an algorithm that solves an instance of T-join problem with m edges by applying a perfect matching algorithm to a graph with $O(m)$ edges. In this manner, we obtain an algorithm for the T-join problem that runs in time roughly $O(n^{3/2})$ (see Theorem 5.6) rather than $O(n^3)$ implied by the method known previously [8], [20]. Finally, we refine this idea to improve the size of the resulting instance of perfect matching (and consequently, the overall running time) by a constant factor.

A. Opportunistic Reductions

In this section we describe simplifying opportunistic reductions that eliminate the nodes of degree 0, 1, and nodes of degree 2 that do not belong to T. These reductions do not improve the worst case performance of algorithms for the T-join problem, but nevertheless help in many real life instances.

The first opportunistic reduction reduces the T-join problem to instances with biconnected graphs.

Theorem 5.1: Consider an instance of the T-join problem described by the graph $\langle V, E\rangle$, edge weight function w and $T \subset V$. Assume that $\langle V, E\rangle$ has biconnected components $\langle V_1, E_1\rangle, \cdots, \langle V_k, E_k\rangle$. Then in linear time we can find sets $T_i \subset V_i$ such that $A \subset E$ is an optimal T-join if and only if for $i = 1, \cdots, k$, $A \cap E_i$ is an optimal T_i-join for $\langle V_i, E_i\rangle$ and $w_{|E_i}$

Proof: If a biconnected component $\langle V_i, E_i\rangle$ happens to be a connected component (or the entire graph) then for obvious reasons it suffices to define $T_i = T \cap V_i$. Now consider $\langle V_1, E_1\rangle$, the first biconnected component reported by Hopcroft's algorithm (see [1, pp. 180–187]); it is a property of this algorithm that this component contains exactly one articulation point, say v. Let $E_0 = E - E_1$, and $V_0 = V - V_1 \cap \{v\}$. We will find sets T_1 and T_0 such that A is a T-join for $\langle V, E\rangle$ if and only if $T_j \cap E_j$ is a solution for $\langle V_j, E_j\rangle$ for $j = 0, 1$. We have four cases. In the first two, $v \in T$. If $|T \cap V_1|$ is even, v must be incident to an odd number of edges from E_1 and, thus, to an even number of edges from E_0. Thus, we can set $T_1 = T \cap V_1$ and $T_0 = T - V_1$. If $|T \cap V_1|$ is odd, then v must be incident to an even number of edges in $A \cap E_1$ and, thus, to an odd number of edges from $A \cap E_0$, consequently we can set $T_0 = T \cap V_0$ and

$T_1 = T - V_0$. In the remaining two cases, $v \notin T$. If $|T \cap V_1|$ is even, v must be incident to an even number of edges from $A \cap E_1$ and an even number from $A \cap E_0$, consequently we can define $T_j = T \cap V_j$ for $j = 0, 1$. If $T \cap V_1$ is odd, v must be incident to an odd number of edges in both $A \cap E_0$ and $A \cap E_1$, so we can define $T_j = T \cap V_j \cup \{v\}$ for $j = 0, 1$. In this fashion, we can each compute T_i as soon as the respective biconnected component $\langle V_i, E_i \rangle$ is reported by Hopcroft's algorithm. □

The second opportunistic reduction eliminates nodes of degree 2 that do not belong to T.

Theorem 5.2: Assume that node $v \notin T$ has exactly two neighbors, v_1 and v_2. Consider the graph transformation where edges $\{v, v_1\}$ and $\{v, v_2\}$ are replaced with edge $\{v_1, v_2\}$ with weight $w(v, v_1) + w(v, v_2)$. Then this edge replacement defines a one-to-one (1–1) correspondence between T-joins of the old graph and the new graph.

Proof: The claim follows immediately from the observation that in the original instance, a solution either contains both e_1 and e_2, or neither of these edges. □

Because the running time of the most efficient algorithms for minimum weight perfect matching depends on the maximum edge weight (if we assume that all weights are integer), we should estimate how much this weight may change. The reduction implied by Theorem 5.1 does not change the maximum edge weight at all, while the reduction implied by Theorem 5.2 increases the maximum by a factor smaller than n.

B. Reduction via Simple Gadgets

We suggest the following algorithm for the T-join problem in sparse graphs.

1) Read T-join instance (V, E, w, T).
2) For each vertex $v \in$ create gadget graph Γ_v that depends only on the degree of v and the membership of v in T. For every neighbor u of v, Γ_v contains a contact node (v, u), all edges of Γ_v have weight zero.
3) For each edge $\{u, v\} \in E$ connect Γ_u and Γ_v with a *replacement* of $\{u, v\}$, $\rho(\{u, v\}) = \{(u, v), (v, u)\}$, that has the same weight.
4) In the resulting graph (V', E', w') find a minimum cost perfect matching M.
5) Return $A = \rho^{-1}(M)$.

Lemma 5.3: 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: Each step of our algorithm, except Step 4, takes linear time; therefore, the running time is determined by the size of the graph (V', E', w') that is produced from our gadgets. The correctness follows from these two properties of our construction.

1) For every T-join $A \subset E$ there exists a set of zero weight edges $Z \subset E'$ such that $\rho(A) \cup Z$ is a perfect matching of (V', E').
2) For every perfect matching $M \subset E'$ the edge set $\rho^{-1}(M)$ is a T-join of (V, E).

Indeed, assume that A' is a minimum cost T-join for the input graph. Then for some zero cost edge set Z, $M = \rho(A) \cup Z$ is a perfect matching of (V', E', w) with the same cost as A'. In turn, $A = \rho^{-1}(M)$ is a T-join with the same cost as M, and by extension, as A'. Therefore, the algorithm returns a T-join with the minimum cost (weight).

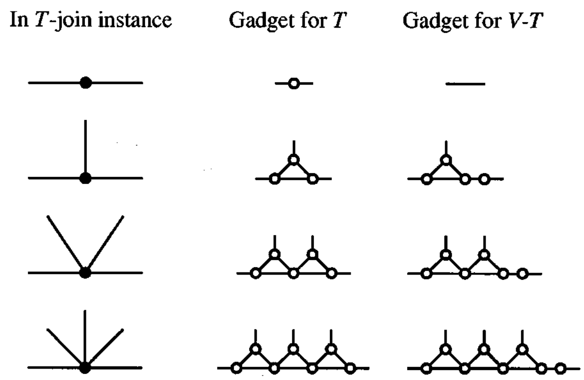


Fig. 8. Replacing a vertex v of the dual graph with a gadget. The *replacement edges* are shown with one endpoint only, *contact nodes* are the nodes that are incident to the replacement edges. Note that the remaining nodes form a horizontal path, and each set of contact nodes can be inserted to this path.

Property 2) 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 1) will be assured as follows. Let M' be the set of replacements of the nodes of A, and let U be the set of nodes that are matched by M'. Consider a gadget Γ_v. Clearly, $\Gamma_v \cap U$ is a set of contacts. The construction of Γ_v assures that for every set of contacts $U\ \Gamma_v - U$ contains a Hamiltonian path of even length and, thus, it contains a perfect matching of zero cost. The union of these perfect matchings form the desired set Z.

To finish the proof, it remains to calculate the size of the new graph. It contains m replacement edges. Moreover, if $v \in T$ has degree d, Γ_v contains $2d - 3$ nodes and $3d - 6$ edges, and if $v \notin T$, then this gadget has one more node and one more edge. The claim follows from the fact that the sum of the node degrees equals $2m$.

Fig. 10 shows an example of a transformation that uses our gadgets; the left part shows an instance of the T-join problem, and the middle part the instance of perfect matching that is obtained with the gadgets from Fig. 8.

C. Reducing T-Join to Perfect Matching with Optimized Gadgets

In this section we describe optimized gadgets which improve the running time by a factor of four on average (see Section VII-B). The construction of these gadgets is less uniform and the connection between the gadget is less regular. We define two versions of gadgets: *full* gadgets, shown in Fig. 9, and *trimmed*. The idea is that the full gadget of node u, denoted Γ_u, contains replacements of all edges that are incident to u; when we connect gadgets of adjacent nodes u and v, we need to decide which replacements of the edge $\{u, v\}$ we remove: those from Γ_u or those from Γ_v? This removal creates

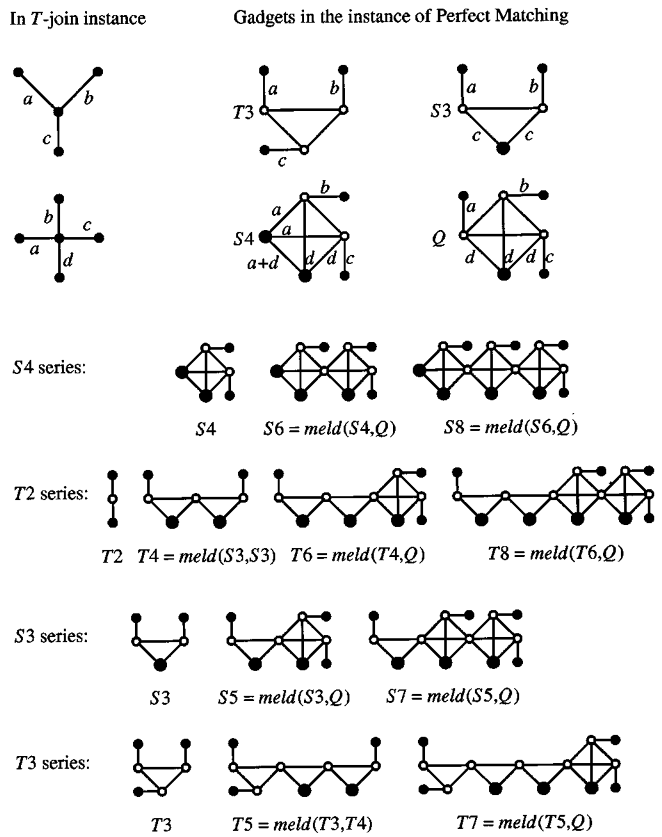
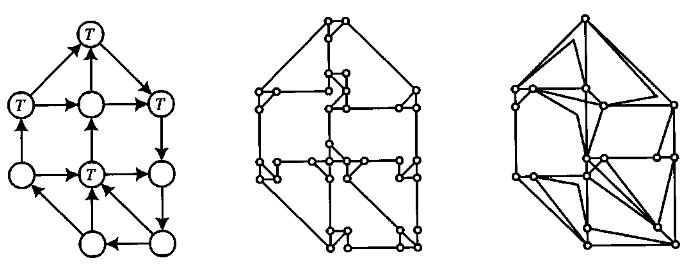


Fig. 9. A better set of gadgets. In the case of the basic gadget, we show explicitly the nonzero edge weights. Empty dots depict the core nodes, and full dots indicate the contacts. Larger full dots indicate obligatory contacts, smaller dots—optional ones.



(a) (b) (c)

Fig. 10. An example of graph transformation. (a) The original graph, a T-join instance, node labels indicate members of T, and arrow heads on edges indicate the direction of the fan out. (b) The perfect matching instance obtained with simple gadgets. (c) An equivalent perfect matching instance obtained with our most "sophisticated." gadgets. Edges that replace the original edges are roughly parallel to the original ones, edges that replace pairs of the original consist of two segments.

the trimmed gadgets which are connected by identifying the respective contact nodes.

More formally, for every node v that is adjacent to u, the gadget Γ_u contains *contact* node (u, v). This contact node is incident to all replacements of the edge $\{u, v\}$ that belong to Γ_u. The remaining nodes of Γ_u are the *core* nodes. We distinguish two kinds of contact nodes: *obligatory* and *optional*. All these kinds of nodes are illustrated in Fig. 9. The property of an optional contact is that it is incident to one edge only, and this edge connects this contact with a core node.

We can trim Γ_u by removing some of its optional contacts. For each contact (u, v), thus, removed we transfer the name (u, v) to its sole neighbor. Now we see that we cannot connect gadgets arbitrarily (as we did in the previous algorithm). In particular, we can connect Γ_u and Γ_v only if one of the respective contacts, (u, v) or (v, u), is optional. Suppose that (u, v) is optional. Then we can connect Γ_u and Γ_v by trimming (u, v), and identifying the new (u, v) with (v, u). As a result, all replacements of edge $\{u, v\}$ are adjacent to a single core node of Γ_u; we say that edge $\{u, v\}$ fans out from u toward v.

Now we can describe the new algorithm.

1) Read instance (V, E, w, T) of T-join problem.
2) For each edge $\{u, v\} \in E$, decide whether it fans out toward u or toward v. Make sure that if a node w has degree d, then at least $\lfloor d/2 \rfloor$ edges are fanned toward w.
3) For each node $u \in V$ create gadget Γ_u that depends on the degree of u and the membership of u in T. For each v adjacent to u, Γ_u contains contact node (u, v). Make sure that if $\{u, v\}$ fans toward v than (u, v) is optional.
4) For each edge $\{u, v\} \in E$ connect Γ_u and Γ_v. In particular, if $\{u, v\} \in E$ fans out toward v, then remove the optional contact (u, v) (and the replacement of $\{u, v\}$ in Γ_u) and identify the new (u, v) with (v, u).
5) In the resulting graph (V', E', w') find a minimum cost perfect matching M using you favorite program.
6) Return A, the set of edges of E that have a replacement in M.

Before we analyze this algorithm, we should note that its step 2) can be implemented in linear time. We apply a graph traversal, where each edge is traversed exactly once, and when we traverse an edge, say from u to v, then we decide to fan it out toward v. If there exists a node adjacent to an odd number of untraversed edges, we start the next stage of the traversal there, and we continue until we encounter another node with this property. When every node is adjacent to an even number of untraversed edges, we can start at any node adjacent to an untraversed edge and continue until we return to this node. It is easy to see that each time we traverse through a node, or when we start and end a traversal in the same node, we fan out one edge toward this node, and one edge away. At most once an edge can be used as a terminal of a noncyclic traversal, thus, we fan at least $\lfloor d/2 \rfloor$ edges toward a node of degree d.

In the new algorithm we have no 1–1 correspondence between edges of (V, E) and their replacements in (V', E'); one edge from E may have up to three replacements, and an edge from E' may replace two edges. However, the relationship is not arbitrary. In particular, because we join gadgets by identifying the respective pairs of contact nodes, the gadgets are edge disjoint and every edge of E' belongs to a gadget; we say that e is a replacement of $\{u, v\}$ if e belongs to Γ_u, the contact (u, v) has not been trimmed and a is incident to this contact. If the other endpoint of e is a core node of Γ_u, then e replaces $\{u, v\}$ only and $w'(e) = w(\{u, v\})$, and if e is also incident to another two contacts, say (u, w), then e is a joint replacement of $\{u, v\}$ and $\{u, w\}$ and $w'(e) = w(\{u, v\}) + w(\{u, w\})$.

It is easy to see that if $e \in E$ and M is a matching of (V', E'), then M contains at most one replacement of e, and that the set of edges from E with replacements in M has the same weight as M. To show the correctness of our algorithm it remains to show

i) For every T-join A of (V, E, w, T) there exists a matching M of (V', E', w') with the same weight; and

ii) for every matching M of (V', E', w'), the set of edges with replacements in M forms a T-join.

We can show that property 2) is assured if every gadget satisfies

ii′) a gadget Γ_u contains an odd number of core nodes if $u \in T$.

Suppose that $u \in M$. Then M contains an odd number of edges with exactly one end point that is a core node of Γ_u; each such edge is a replacement of one edge incident to u. Moreover, M may contain edges that are adjacent to the contacts of Γ_u, but not to the core nodes; each such edge is a replacement of two edges adjacent to u. Taken together, the number of edges with a replacement in M that are incident to u is odd. Similarly, if $u \notin T$, this number is even and, therefore, the set of edges with replacements in M forms a T-join.

Property 1) is somewhat more complicated to verify. Suppose that A is a T-join, and consider a full gadget Γ_u. If an edge $\{u, v\} \in A$, then we have to find a match of (u, v) inside Γ_u. One can see that 1) will follow if every gadget satisfies

i′) every induced subgraph of Γ_u that contains all core nodes and has even number of nodes contains a perfect matching.

We can reformulate the latter condition to make it easier to prove. Suppose that we have an induced subgraph of a gadget that contains all the core nodes and has even number of nodes. Suppose that this subgraph contains an optional contact. This node has only one neighbor, so the subgraph in question contains a perfect matching if and only if it contains one after we remove this optional contact and its sole neighbor.

With this motivation, we define the reduced form of a gadget as one where all optional contacts are removed, and the former neighbors of these contacts are now viewed as contacts, rather than the core nodes. Now condition i′) reduces to

i″) every induced subgraph of a reduced gadget that contains all core nodes and has even number of nodes contains a perfect matching.

The simplest way property i″) can be assured if when the reduced gadget is a clique: an induced subgraph of a clique is a clique, and a clique of even size contains a perfect matching. Our *basic gadgets* have exactly this form.

To construct other provably correct gadgets, we will introduce some terminology. We use Ti to denote a gadget for a T-node of degree i, and Si to denote a gadget for a $(V - T)$-node of degree i. The reduced forms of our basic gadgets, namely, of $S3$, $T3$, $S4$ and Q (a version of $S4$), are cliques, so these gadgets are proven correct. For larger gadget, we define them, and prove their correctness, inductively.

Given two gadgets H and H', we can *meld* them as follows. Let x be an optional contact of H and x' be an optional contact of H'. Let y and y' be the core nodes adjacent to x and x'. Then $meld(H, H')$ is created by discarding x and x', and by identifying y with y'. Fig. 9 shows many examples of melding. For $i \geq 7$ we define Si as $meld(S(i-2), Q)$ and Ti as $meld(T(i-2), Q)$. The following lemma validates building larger gadgets by melding the smaller ones.

We will say that a graph is a gadget if it has a distinguished set of core nodes, and every superset of the core nodes of even size has a perfect matching. An S gadget has an even number of core nodes, and a T gadget has an odd number. An i gadget has i noncore nodes.

Lemma 5.4: We can build new gadgets in the following three ways:

i) If H is an Si gadget, and H' is an Sj, then $meld(H, H')$ is a $T(i+j-2)$.

ii) If H is an Ti gadget, and H' is an Tj, then $meld(H, H')$ is a $T(i+j-2)$.

iii) If H is an Si gadget, and H' is an Tj, then $meld(H, H')$ is a $S(i+j-2)$.

Proof: Let H_0 denote $meld(H, H')$. First note that H_0 contains all the contacts of H and H' except for the two that are discarded in the process of melding. Thus, H_0 has $i+j-2$ contact nodes. Second, the number of core nodes of H_0 is the sum of the respective numbers for H and H' minus one (because we replaced two core nodes with one). Thus, we can verify that H_0 satisfies property ii′). It remains to show that H_0 satisfies property i″), i.e., that every subset of its reduced form that contains all the core nodes and has even size possesses a perfect matching.

Note that the "fused node" v was a contact in the reduced forms of H and H', but is not a contact in the reduced form of H_0, because it is not adjacent to an optional contact anymore. Consider now an induced subgraph of H_0 with set of nodes U_0, that contains all the core nodes and has even size. Let U be the intersection of $U_0 - \{v\}$ with the reduced form of H, and U' be the intersection of $U_0 - \{v\}$ with the reduced form of H'. Clearly, U_0 is a disjoint union of U, U', and $\{v\}$. Because $|U_0|$ is even, either $|U|$ is odd and $|U'|$ is even, or vice versa. Assume the former. Then $U \cup \{v\}$ contains a perfect matching inside H and U' contains a perfect matching inside H', so there exists a perfect matching for U_0.

The efficiency of the improved algorithm can be expressed with the following theorem.

Theorem 5.5: Consider an instance of minimum cost T-join problem with n nodes, m edges, n_0 T nodes of odd degree and n_1 nodes of T that have degree 3. Our algorithm solves this instance by generating an equivalent instance of the minimum cost perfect matching that has at most $2m - n + n_0$ nodes and at most $6m - 5.5n + 0.5n_1$ edges.

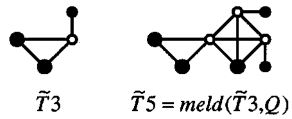


Fig. 11. More efficient forms of $T3$ and $T5$. Note that they have more obligatory contacts than the optional ones.

Proof: Fig. 9 shows both the basic gadgets, and the remaining ones that are formed with melding. For $i \geq 5$ we define $S(i+2)$ as $meld(Si, Q)$ and $T(i+2)$ as $meld(Ti, Q)$. Note that a gadget for a node of degree d has at most $d-1$ core nodes, except for the T nodes of odd degree; if the number of edges is m, number of nodes n and the number of odd degree T-nodes is n_0, this leads to at most $m - n + n_0$ nodes in our (V', E', w') graph. Similarly, the number of edges in an Xd gadget is at most $3.5d - 5.5$, with the exception of T nodes of degree 3, which have 0.5 more edges in their gadgets. Because each of the "original" edges is counted twice, this yields the upper bound of $6m - 5.5n + n_1$ edges in (V', E', w'). □

Finally, we can apply the best known so far algorithm by Gabow and Tarjan [6] to solve the perfect matching problem.

Theorem 5.6: There exists an algorithm that solves the minimum perturbation problem in time $O((n \log n)^{3/2})\alpha(n)$, where α is the inverse Ackerman function.[6]

D. Further Work

Can we obtain an optimum set of gadgets for the reduction of T-join to perfect matching? The answer is complicated, because one would have to make certain "natural" assumption about the gadgets. In our initial attempts, we assumed that gadgets replacing nodes of the same degree are identical. However, later we reduced the average gadget size by violating this assumption (i.e., by assuming that the majority of edges incident to some node of odd degree is directed toward this node). Fig. 11 shows alternative forms of certain gadgets (i.e., $T3$ and $T5$ that arise under such an alternate assumption), and Fig. 10 shows a perfect matching instance obtained with such gadgets. We believe that this topic merits further investigation.

VI. Approximation Algorithms for the T-Join Problem

The exact algorithm from the previous section may still be too slow for processing very large layouts. A naive method, used by Ooi *et al.* [19], 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 nonbipartite 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

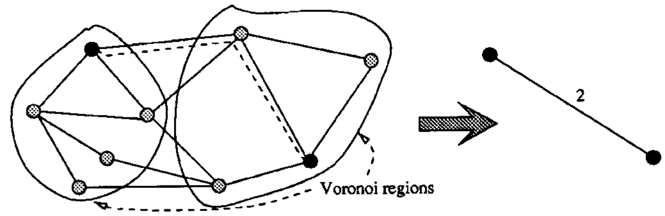


Fig. 12. The geometric dual graph G and the Voronoi graph for T. The weight of an edge in $Vor(G)$ is the total cost of edges in the shortest path (dashed edges) between the centers of the Voronoi regions (black vertices).

edges with the same phase assigned to both ends can be several times larger than for the exact algorithm (see Section VII-B).

We suggest a fast heuristic based on the Voronoi graph paradigm of [14].

A. Iterated Voronoi Algorithm

1) Partition all nodes in G into the *Voronoi regions* of the nodes in T [14] such that:
   a) each Voronoi region contains exactly one node from T, which is the *center* of the Voronoi region, and
   b) each even degree node belongs to the Voronoi region that contains the closest node in T (break ties arbitrarily).
2) Construct the *Voronoi graph* $Vor(G)$ in which each node represents a Voronoi region, and two nodes R_1 and R_2 are adjacent if and only if a shortest weighted path in G between the corresponding region centers is completely contained in the two regions (see Fig. 12).
3) Find a minimum weight maximum matching M in $Vor(G)$.
4) 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 $Vor(G)$.
5) Delete edges VM from the conflict graph G.
6) Repeat previous steps while $T \neq \emptyset$.

We have also implemented a well-known 2-approximation algorithm suggested by Goemans and Williamson in [7] (see Section VII-B). This algorithm is rather simple because it does not rely on solving the minimum weight perfect matching problem. Unfortunately, it is mostly intended to handle *dense* cases of the T-join problem and its runtime is slightly worse than the runtime for the optimal algorithm based on optimized gadgets (see Section V-C).

VII. Implementation Experience

A. Software Implementation

We have implemented the "one shot" method in C++ on the Solaris 2.6, Sun CC 4.2 platform. Input is (hierarchical) GDSII that is converted to CalTech Interchange Format (CIF), then read into an internal polygon database[7] There are two major software elements: the phase generator and the graph based compactor.

[6]While the definition of $\alpha(n)$ is complex, it suffices to know that $\alpha(n) \leq 4$ for $n < 2^{197}$.

[7]Our implementation is currently restricted to rectilinearly oriented features, but there are no major obstacles to handling octilinear or all angle geometries (e.g., slicing would be into parallelograms or trapezoids, respectively). The focus of our work is on fast optimal solution of sparse (planar) T-join instances; this is the key to phase shiftable layout design no matter what angles on features are allowed.

TABLE I
COMPUTATIONAL RESULTS FOR PHASE ASSIGNMENT OF LAYOUTS WITH VARIOUS SIZES

Testcases	Layout1		Layout2		Layout3	
#polygons/#edges	3769	12442	9775	26520	18249	51402
Algorithms	#conflicts	runtime	#conflicts	runtime	#conflicts	runtime
Greedy[19]	2650	.58	2722	3.45	6168	4.95
Iterated Voronoi	1828	2.16	1552	4.99	3494	12.43
Simple Gadgets	1468	18.78	1346	12.49	2958	60.49
Optimized Gadgets	1468	3.62	1346	5.17	2958	16.05
GW[7]	1612	3.79	1488	5.18	3280	17.90

The top row gives the number of polygons and the number of conflict edges for each testcase. The bottom five 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 phase assignment algorithms suggested in [2], [12], and [19], a method based on approximation algorithm by Goemans–Williamson [7] and the present paper. All runtimes are in seconds for a 300-MHz Sun Ultra-10 workstation with 128-MB RAM.

Below, we describe the implementation of these two parts, and in the next two subsections we present results of the computational experience for the phase generator and for the entire flow including the compactor.

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) [5], 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. [In all the results below, we use $B = 2b$, with b set to the minimum wire width on the M1 and M2 layouts that we compact.[8] Layout area is computed based on an assumed 200 nm (0.2 micron) minimum wire width.]

The *graph based compactor*: 1) generates constraints between feature edges according to standard design rules and an efficient sweepline approach, 2) adds constraints produced by the phase generator, and 3) 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 [3]. We consider three types of constraints: 1) *shape* constraints fix the shape of features which cannot be changed according to design rules; 2) *overlap* constraints ensure that features will remain electrically connected after compaction, and/or properly aligned between different layers; and 3) *spacing* constraints enforce separation design rules (including PSM specific separation rules between features of the same or of the different phase).[9]

Note that we perform y-compaction the same way that we do x-compaction (after temporarily swapping x and y coordinates). In the one shot flow, we output the new positions of all features after compacting exactly once in each of the x and y directions.

[8]No dark field PSM rules for poly are available, and our compactor is not able to handle nonrectilinear shapes commonly found on poly. Thus, for the proof of concept described below, we use local metal layouts.

TABLE II
COMPUTATIONAL RESULTS FOR COMPACTION WITH DIFFERENT PHASE ASSIGNMENT FOR LAYOUTS WITH VARIOUS SIZES

Testcases	Layout1		Layout2		Layout3	
#polygons	1416		3339		6013	
Phase Assignment	#conflicts	area	#conflicts	area	#conflicts	area
Ideal (w/o conflicts)	0	.36641	0	1.7106	0	2.0712
Optimal	94	.43583	112	1.9257	220	2.2954
Greedy	246	.47339	168	2.0283	238	2.3289

The top row gives the number of polygons for each testcase. The bottom three rows contain the numbers of unresolved conflict edges (i.e., the numbers of pairs of polygons with additional psm constraint imposing distance at least B between features with the same phase) and the compacted layout area for different phase assignments. Phase assignments are ideal (when we assume that no phase conflicts are left unresolved), optimal (when the minimum possible number of phase conflicts are left unresolved), and greedy (when phase assignment greedily minimizes unresolved conflicts [19]). All areas are in $10^3\mu^2$, Assuming a 200-nm (0.2 μm) minimum feature width in the process.

B. Computational Experience with the Phase Generator

Table I 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 128-MB 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 I 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 [12] and [19]: it reduces by 40% the number of unresolved conflicts, which correspondingly reduces the amount of layout modification needed in compaction. Finally, we also implemented the approximation algorithm for the T-join problem from [7]. Our results show that the average deviation from the optimum for the Goemans–Williamson algo-

[9]We have implemented compaction solely to verify that improved odd cycle breaking can result in reduced compacted layout area. Our compactor is not fully functional, e.g., it does not handle all angle geometries and it has limited capacity. On the other hand, in Section VII-C below we present results showing that even with this limited functionality, and without any link between the compaction graph and conflict edge weighting, we still obtain better results from improved odd cycle breaking.

rithm is around 10%, which is significantly larger than the 2% for Euclidean matchings reported in [23].

C. Computational Experience with the Graph Based Compactor

Table II summarizes the results of experiments with the graph based compactor on three different designs. The layouts for which we report compaction results are small, again because of the limited capacity of our prototype compactor and a lack of constraint pruning in the compaction graph. The compaction is performed over four layers: METAL2, VIA, METAL1, and CONTACT. We first compact the design without imposing PSM constraints.[10] In other words, the resulting area is the absolute minimum that can be achieved with phase shifting technology, since we assume that the phase assignment will resolve all conflicts (i.e., that such a phase assignment exists). Next, we perform compaction with phase assignment: 1) having the optimal number of unresolved conflicts or 2) induced by the greedy method. Table II clearly shows the dependency of area on the number of unresolved conflicts. The greater the ratio of unresolved conflicts in the greedy method, compared to the optimal number of unresolved conflicts, the more the optimal method gains in area over the greedy method.

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. Our experience with the graph based compactor following phase assignment shows that significant potential area reductions (around 10%) may be achievable versus previous conflict breaking techniques. This is the case even though our conflict breaking does not exploit any available knowledge of critical paths in the compaction graph. Other, potentially more powerful approaches to layout modification and phase assignment for alternating PSM—as well as stronger compaction techniques—are currently under investigation. We are also pursuing unified solutions to the dark field and bright field alternating PSM contexts.

ACKNOWLEDGMENT

The authors would like to thank A. Caldwell, S. Mantik, and I. Markov for critical assistance in their software development.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ulman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison Wesley, 1974.

[2] P. Berman, A. B. Kahng, D. Vidhani, H. Wang, and A. Zelikovsky, "Optimal phase conflict removal for layout of dark field alternating phase shifting masks," presented at the ACM/IEEE Int. Symp. Physical Design, 1999, to appear.

[3] C. Bamji and R. Varadarajan, *Leaf Cell and Hierarchical Compaction Techniques*. Boston, MA: Kluwer Academic, 1997.

[4] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*, New York: Wiley Interscience, 1998.

[5] W. Cook and A. Rohe, "Computing minimum-weight perfect matchings," unpublished, August 1998.

[6] H. N. Gabow and R. E. Tarjan, "Faster scaling algorithms for general graph matching problems," *J. ACM 38*, pp. 815–853, 1991.

[7] M. X. Goemans and D. P. Williamson, "A general approximation technique for constrained forest problems," *SIAM J. Computing 24*, pp. 296–317, 1995.

[8] F. O. Hadlock, "Finding a maximum cut of a planar graph in polynomial time," *SIAM J. Computing*, vol. 4, no. 3, pp. 221–225, 1975.

[9] A. B. Kahng, G. Robins, A. Singh, H. Wang, and A. Zelikovsky, "Filling and slotting: Analysis and algorithms," *Proc. ACM/IEEE Int. Symp. Physical Design*, pp. 95–102, 1998.

[10] A. B. Kahng, G. Robins, A. Singh, and A. Zelikovsky, "Filling algorithms and analyzes for layout density control," IEEE Trans Computer-Aided Design, to be published.

[11] A. B. Kahng and H. Wang, "Toward lithography aware layout: Preliminary litho notes," unpublished, July 1997.

[12] A. B. Kahng, H. Wang, and A. Zelikovsky, "Automated layout and phase assignment techniques for dark field alternating PSM," in *Proc. SPIE 11th Annu. BACUS Symp. Photomask Technology*, SPIE 1604, 1998, pp. 222–231.

[13] M. D. Levenson, N. S. Viswanathan, and R. A. Simpson, "Improving resolution in photolithography with a phase shifting mask," *IEEE Trans. Electron Devices*, vol. ED-29, pp. 1828–1836, Nov. 1982.

[14] K. Mehlhorn, "A faster approximation algorithm for the Steiner problem in graphs," *Inform. Processing Lett.*, ser. 27, pp. 125–128, 1988.

[15] A. Moniwa, T. Terasawa, N. Hasegawa, and S. Okazaki, "Algorithm for phase shift mask design with priority on shifter placement," *Jpn. J. Appl. Phys.*, vol. 32, pp. 5874–5879, 1993.

[16] 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.*, vol. 34, pp. 6584–6589, 1995.

[17] J. Nistler, G. Hughes, A. Muray, and J. Wiley, "Issues associated with the commercialization of phase shift masks," in *Proc. PIE 11th Annu. BACUS Symp. Photomask Technology*, SPIE 1604, 1991, pp. 236–264.

[18] K. Ooi, S. Hara, and K. Koyama, "Computer aided design software for designing phase shifting masks," *Jpn. J. Appl. Phys.*, vol. 32, pp. 5887–5891, 1993.

[19] K. Ooi, K. Koyama, and M. Kiryu, "Method of designing phase shifting masks utilizing a compactor," *Jpn. J. Appl. Phys.*, vol. 33, pp. 6774–6778, 1994.

[20] G. I. Orlova and Y. G. Dorfman, "Finding the maximum cut in a graph," *Engr. Cybern.*, vol. 10, pp. 502–506, 1972.

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

[22] SIA, *The National Technology Roadmap for Semiconductors*, Semiconductor Industry Association, San Jose, CA, Dec. 1997.

[23] D. P. Williamson and M. X. Goemans, "Computational experience with an approximation algorithm on large scale Euclidean matching instances," *INFORMS J. Computing*, vol. 8, pp. 29–40, 1996.

Piotr Berman received the M.S. degree in mathematics at the University of Warsaw, Poland, in 1978 and the Ph.D. degree in applied mathematics from Massachusetts Institute of Technoogy. Cambridge, in 1985.

He is an Associate Professor at the Department of Computer Science and Engineering of the Pennsylvania State University, University Park. He has worked at Penn State since, except for visiting positions at the Universities of Chicago (Il), Bonn, (Germany), Tel-Aviv, (Israel), and Lund, (Sweden). His current research interests are combinatorial optimization problems and their applications in computational biology, CAD, and scheduling. In the last three years he has coauthored nine journal papers and 11 conference/workshop papers.

[10]During this compaction we set the minimum separation B between same phase features equal to the minimum separation b between opposite phase features.

Andrew B. Kahng received the A.B. degree in applied mathematics (physics) from Harvard College, Cambridge, MA. He received the M.S. and Ph.D. degrees in computer science from the University of California, San Diego.

From June 1983 to June 1986, he was with Burroughs Corporation Micro Components Group, San Diego, CA, where he worked in device physics, circuit simulation, and CAD for VLSI layout. He joined the University of California, Los Angeles, Computer Science Department as an Assistant Professor in July 1989, and became an Associate Professor in July 1994 and Professor in July 1998. From April 1996 to September 1997, he was on sabbatical leave and leave of absence from UCLA, as Visiting Scientist at Cadence Design Systems, Inc. He resumed his duties at UCLA in Fall 1997, and presently serves as the Computer Science Department's Vice Chair for graduate studies. He currently has many active working relationships with industry. His interests include VLSI physical layout design and performance analysis, combinatorial and graph algorithms, and stochastic global optimization.

Prof. Kahng has received the National Science Foundation (NSF) Research Initiation and Young Investigator Awards, and a DAC Best Paper Award. He was the Founding General Chair of the 1997 ACM/IEEE International Symposium on Physical Design, and defined the physical design roadmap as a member of the Design Tools and Test Technical Working Group for the 1997 renewal of the SIA National Technology Roadmap for Semiconductors. He is currently a member of the EDA Council's EDA 200X task force, and the design technical working group for the 1999 SIA ITRS renewal.

Devandra Vidhani, received the B. Tech. degree in computer science and engineering from the Indian Institute of Technology, Kanpur, in 1995, and the M.S. degree in computer science from the University of California, Los Angeles, in 1998.

From June 1995 through October 1997, he was a Member of Technical Staff at Cadence Design Systems (I). Since April 1999, he is with Sun Microsystems Inc. as a Senior CAD Engineer. His research interests include layout parasitics extraction and noise and signal integrity issues.

Huijuan Wang received the B.S. degree in chemistry from the University of Science and Technology of China, Beijing, in 1995, and the M.S. degree in computer science from the University of California, Los Angeles in 1998

She is currently a Member of Technical Staff at Sun Microsystems, Inc., Microelectronics division. Her research interests include physical design automation and interconnect parasitic extraction methodology.

Alexander Zelikovsky received the A.B. and M.S. degrees in mathematics from Kishinev University, Moldova, In April 1989, he received the Ph.D. degree in computer science from the Institute of Mathematics of the Belorussian Academy of Sciences, Minsk, Belarus.

From 1982 to October 1985, he was with the Institute of Mathematics of Moldova Academy of Sciences in Kishinev, where he worked in system programming. He was a Senior Research Scholar with the Institute of Mathematics in Kishinev until September 1995. In 1992–1995, he participated in Volkswagen Stiftung project and was on leave of absence from the Institute of Mathematics as Visiting Researcher at the Computer Science Department of Bonn University, Bonn, Germany and at Institut fur Informatik (Saarbrueken, Germany). From 1995 to 1997, he was a Research Scientist at the University of Virginia, Charlottesville, and from 1997 to 1999, he was a Postdoctoral Scholar at the University of California, Los Angeles. Since January 1999, he has been an Assistant Professor at the Computer Science Department of Georgia State University, Atlanta. He has authored more than 40 refereed publications. His research interests include discrete and approximation algorithms, computational geometry, VLSI physical layout design and performance analysis; and combinatorial optimization and computational biology.