The T-join Problem in Sparse Graphs: Applications to Phase Assignment Problem in VLSI Mask Layout*

Piotr Berman[1], Andrew B. Kahng[2], Devendra Vidhani[2], and Alexander Zelikovsky[3]

[1] Dept. of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802-6106, berman@cse.psu.edu
[2] Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90095-1596, {abk,vidhani}@cs.ucla.edu
[3] Department of Computer Science, Georgia State University, University Plaza, Atlanta, GA 30303, alexz@cs.gsu.edu

Abstract. Given a graph G with weighted edges, and a subset of nodes T, the T-join problem asks for a minimum weight edge set A such that a node u is incident to an odd number of edges of A iff $u \in T$. We describe the applications of the T-join problem in sparse graphs to the phase assignment problem in VLSI mask layout and to conformal refinement of finite element meshes. We suggest a practical algorithm for the T-join problem. In sparse graphs, this algorithm is faster than previously known methods. Computational experience with industrial VLSI layout benchmarks shows the advantages of the new algorithm.

1 Introduction

Given a graph G with weighed edges, and a subset of nodes T, the *T-join Problem* seeks a minimum weight edge set A such that a node u is incident to an odd number of edges of A iff $u \in T$. One can find a discussion of the T-join problem in Cook *et al.* [4], pp. 166-181.

In this work, we develop a new exact algorithm for the T-join problem which is motivated by the applications in VLSI mask layout. Section 2 describes the context of the phase assignment problem in VLSI phase-shifting masks. The corresponding graphs are sparse, with a large number (up to millions) of nodes. Similar graphs appear in conformal refinement of finite element meshes.

A traditional reduction of the T-join problem to minimum weight perfect matching is too time- and memory-consuming to be practical. In Section 3 we suggest a new reduction to the perfect matching problem which increases the size of the graph by at most a factor of two. This reduction is linear and does not contain any large hidden constants. The achieved runtime is $O((n \log n)^{3/2})\alpha(n)$,

* This work was supported by a grant from Cadence Design Systems, Inc. P. Berman was partially supported by NSF Grant CCR-9700053 and A. Zelikovsky was partially supported by GSU Research Initiation Grant #00-013.

where α is the inverse Ackerman function and n is the number of nodes in G. In the concluding Section 4 we describe our computational experience with layouts derived from standard-cell VLSI designs obtained from industry.

2 Phase Assignment in VLSI Phase Shifting Masks

In the manufacture of a given VLSI circuit layout, photosensitive material is exposed to light that is passed through a *mask*. Without loss of generality, *clear regions* in the mask correspond to desired shapes, or *features*, in the layout. *Phase-shifting mask* (PSM) technology, proposed by Levenson et al. [11] in 1982, enables the clear regions of a mask to transmit light with prescribed phase shift. Given two adjacent clear regions with small separation, and respective phase shifts of 0 and 180 degrees, the light diffracted into the nominally dark region between the clear regions will interfere *destructively*; this gives improved image contrast (i.e., between light and dark) and better resolution of the two features. PSM is enabling to the subwavelength optical lithography upon which the next several VLSI process generations depend [17].

Two positive constants $b < B$ define a relationship between manufacturability of the layout and the distance between any two given clear regions [15]. The distance between 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] which can be resolved by assigning opposite phases to the conflicting features. In other words, B defines the minimum spacing when two features have the same phase, while b defines the minimum spacing when the features have opposite phases. If the distance between two features is greater than B, there is no phase conflict and any phase assignment is allowed.

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

Given a layout, consider the *conflict graph* $G =< V, E >$ which has a vertex for each feature, and an edge between two vertices iff the corresponding features are in phase conflict. Observe that the Phase Assignment Problem can be solved iff the conflict graph is bipartite, i.e., has no odd cycles. The only way to change the conflict graph is to perturb the layout, e.g., perturb the locations of features such that they are no longer in phase conflict. Thus, if the conflict graph is not bipartite, we seek a minimal perturbation of the layout such that the conflict graph in the new layout is bipartite. The following method for layout modification and phase assignment was proposed in [10], extending work of [15].

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

[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.

(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., apply a layout compaction tool that enforces separation at least B between features that are assigned the same phase, and separation at least b between features that are assigned different phases.

In this approach, the key step is determining which set of edges in the conflict graph correspond to a "minimum perturbation" of the layout.

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.

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 $G =< V, E >$ 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, than an edge of G connects f with g if it belongs to both of them. A *reduced dual* of G is a graph $\bar{D} =< F, \bar{E} >$ 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.

Lemma 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 Figure 1). The odd faces of G form odd-degree vertices 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 $< F, A >$. 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 iff each connected component of $< F, A >$ contains an even number of odd nodes (odd faces of G). Moreover, for each feasible solution $A \subset E$ there exists a feasible solution $\bar{A} \subset \bar{E}$ with weight that is not larger; we obtain $\bar{A}$ 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 $\bar{D}$. ❑

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 1), starting from arbitrary vertex v breadth-first search 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, u is assigned the opposite phase. Such breadth-first search can be performed in linear time.

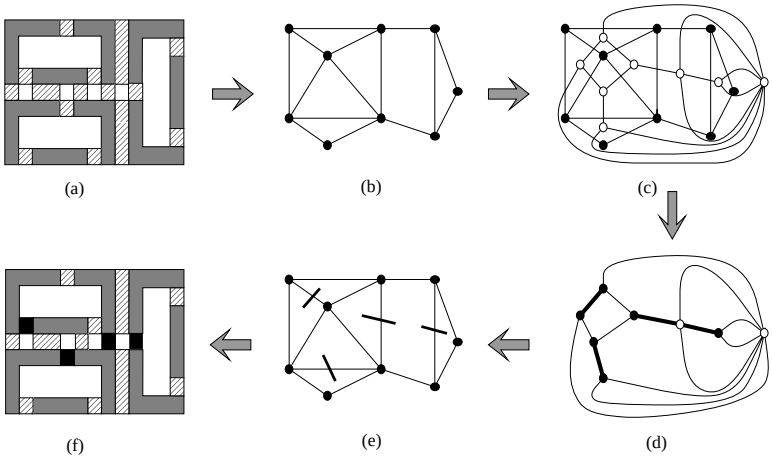

Fig. 1. From the conflicts between features (a), the conflict graph is derived (b). The dual graph (c) is constructed. The vertices of odd degree are matched using paths in the dual graph (d), and the corresponding conflict edges are determined (e). Finally, the minimum set of conflicts to be deleted is determined (f).

Quadrangulations for Finite Element Meshes

Another application of the sparse T-join Problem is described in [14]. Conformal mesh refinement has gained much attention as a necessary preprocessing step for the finite element method in the computer-aided design of machines, vehicles, and many other technical devices. For many applications, such as torsion problems and crash simulations, it is important to have mesh refinements into quadrilaterals. The problem of constructing a minimum-cardinality conformal mesh refinement into quadrilaterals is well known. This problem is NP-hard and for meshes without so-called folding edges a 1.867-approximation algorithm is suggested in [14]. This algorithm requires $O(nm \log n)$ time, where n is the number of polygons and m the number of edges in the mesh. The asymptotic complexity of the latter algorithm is dominated by solving a T-join problem in a certain variant of the dual graph of the mesh. Although the T-join problem can be solved fast for planar graphs by an application of the planar separator theorem (see [12] and [2]), our reduction is much simpler and does not contain any large hidden constants.

3 A Fast Algorithm for the T-join Problem

The T-join problem was solved by Hadlock [8] and Orlova & Dorfman [16] using the following reduction.

Lemma 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 2 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 Minimum-Weight Perfect Matching Problem may have many more edges than necessary, and thus itself is too difficult to be used in practice. The present work provides 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).

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

3.1 Opportunistic Reductions

In this subsection we describe simplifying, "opportunistic" reductions that serve to normalize input graphs for the reduction from the T-join problem to perfect matching that is described later. 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 1 *Consider an instance of the T-join problem described by the graph $< V, E >$, edge weight function w and $T \subset V$. Assume that $< V, E >$ has biconnected components $< V_1, E_1 >, \ldots, < V_k, E_k >$. 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 $i = 1, \ldots, k$, $A \cap E_i$ is an optimal T_i-join for $< V_i, E_i >$ and $w_{|E_i}$.*

Proof. If a biconnected component $< V_i, E_i >$ happens to be a connected component, then for obvious reasons it suffices to define $T_i = T \cap V_i$. Similarly, the claim is trivial if $< V, E >$ is biconnected. Now consider $< V_1, E_1 >$, 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 $< V, E >$ if and only if $T_j \cap E_j$ is a

solution for $< V_j, E_j >$ 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 adjacent 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 $< V_i, E_i >$ is reported by Hopcroft's algorithm. ❑

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

Theorem 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 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 1 does not change the maximum edge weight at all, while the reduction implied by Theorem 2 increases the maximum by a factor smaller than n.

3.2 Reducing T-join to Perfect Matching with Gadgets

In this subsection, we develop a new and more efficient reduction of the T-join problem to perfect matching, using gadgets. The general outline of our reduction is the following. For each instance (G, w, T) of the T-join problem we will construct an equivalent instance (G', w') of the perfect matching problem. Each node v will be replaced with a *gadget* graph G_v that is connected with edges of weight 0. Later we will call these edges *connections*. Each edge $\{u, v\}$ will be replaced with 1, 2, or 3 edges that connect G_u with G_v. We will call these edges *replacements*. Each replacement has the same weight as the corresponding original edge.

From the previous subsection, we may assume the following restrictions on instances of the T-join problem: the graph is biconnected, $|T|$ is even and positive, all T-nodes have degree at least 2 and all other nodes have degree at least

3. Henceforth, we will use S to denote $V - T$. The correctness of the translation is assured by the following *strong equivalence* condition. For each perfect matching M in G' there exists a solution A in G with 1-1 correspondence between replacements in M and edges in A where each edge $e \in A$ corresponds to one of its own replacements. Conversely, for every solution A in G there exist a perfect matching M in G' with such correspondence.

We will assure the strong equivalence using the following lemma.

Lemma 3 *Properties (1), (2) and (3) are sufficient to assure strong equivalence between G and G':*

(1) For any edge $\{u, v\}$, there is a node which is incident to all replacements of $\{u, v\}$. If this node belongs to G_v, then we say that $\{u, v\}$ fans out from v towards u.

(2) If $u \in T$ then G_u contains an odd number of nodes, and if $u \in S$ then G_u contains an even number of nodes.

(3) Let A_u be a set of edges that are incident to some node u of the graph G. Assume that $|A_u|$ has the same parity as $|G_u|$. Then all nodes of G_u are included in a matching M_u that consists only of the replacements of the edges of A_u and the connections of G_u.

Proof. Given a matching M in G', we obtain a corresponding T-join A in G by discarding all connections, and exchanging the replacements for the "original" edges. Property (1) assures that there is a 1-1 correspondence between replacements in M and edges in A since only one edge from replacement can participate in a matching. Note that each replacement can match at most one node in G_u and connections in G_u can match only even number of nodes. Thus property (2) assures that if $u \in T$, then A contains an odd number of edges incident to u, and if $u \in S$, then A contains an even number of such edges.

It remains to show a converse relationship. Consider a T-join A in G. By property (3), we can find a matching M_u for every group of edges G_u, so it remains to show that we can combine these matchings together. By property (1), every edge $\{v, u\}$ with more than one replacement has all its replacements incident to a single node; if this node is in G_v, then this edge fans out from v toward u. Let us remove from each M_u the replacements of edges that fan out from u and take the union of the reduced M_u's. If a node w is not matched, it must belong to some G_u, and its incident edge from M_u was removed, because it was a replacement of an edge that fans out from u toward some v. However, in this case, $\{u, v\} \in A$ and one of the replacements of this edge must still belong to M_v, moreover, it must be incident to w. ❑

Properties (1) and (2) can be immediately verified for a given construction of graph G'. Property (3) will be proven by induction on the degree of u. However, as a preliminary step, we must show that we can simultaneously provide a gadget G_v for every node v of G. The limitation is that almost every gadget requires that a certain number of edges adjacent to v fan out toward v. However, this requirement never exceeds half of the total number of incident edges (degree of the node). Thus, before proceeding further, we should show that

Lemma 4 *We can fan out the edges of G in such a way that if a node has degree $2k$ or $2k+1$, at least k edges are fanned out toward it (see Figure 2).*

Proof. By induction on number of edges in G. We first consider the case when G contains a simple cycle. Then we fan out the edges on this cycle so each is fanned out toward a different node, and remove the edges of this cycle. For each affected node, the degree decreases by 2, and the number of edges that fan toward such a node decreases by 1.

If G contains no cycles, then it is a forest; we can take an edge that is incident to a leaf, fan it out toward its other end and remove this edge. Two nodes are affected: a leaf which does not have any requirements ($k = 0$), and the other node, for which the degree decreased by 1, so that the requirement for more edges fanned out toward it decreased by 1 as well. ❑

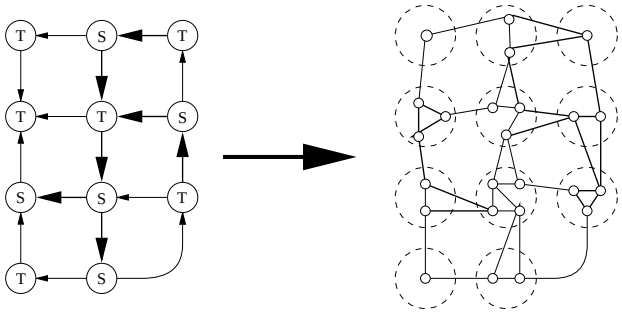

Fig. 2. An example of graph transformation. In the original graph, node labels indicate the member of T and S respectively, arrows on edges indicate the direction of the possible fan out. The thick edges will be fanned out during the transformation.

The gadgets that we use are formed from three kinds of building blocks. A gadget for node v is defined as the graph H that consists of G_v, plus the adjacent replacement edges. If the replaced edge fans out toward v, we keep all the replacements, and otherwise we keep only one. Because we define gadgets first, and assign them to various nodes later, we will use $core(H)$ to denote G_v, and $rind(H)$ to denote $H - core(H)$. In general, a gadget H is characterized by the degree of its node and by the membership of this node in T or S. We will use acronyms to identify gadgets, e.g., $T4$ are gadgets for elements of T that have degree 4.

We can now formulate the sufficient conditions for the correctness of a gadget that are implied by Lemma 3. Property (1) of Lemma 3 is assured as follows: a Ti or Si gadget has $|rind(H)| = i$; each edge incident to the node represented by the gadget corresponds to one of the nodes of $rind(H)$; if this edge fans out toward that node, this node is connected with $core(H)$ by all its replacements, otherwise it is connected with $core(H)$ by a single edge.

Property (2) is assured simply if $|core(H)|$ is odd for a T gadget, and even for an S gadget. Finally, property (3) means that if $core(H) \subset U \subset H$ and $|U|$ is even, then H contains the matching that matches all nodes of U and no other nodes.

Now we will describe construction of the gadgets. We first define three basic gadgets, $S3$, $T3$ and Q, which is actually a variety of $T4$ (see Fig. 3). Gadget $T2$ is a degenerate case, because we do not modify T-nodes of degree 2 (except than an edge originating in such node may fan out toward its other end). Fig. 3 shows how we form gadgets for all nodes of degree below 7. For nodes of degree more than 6, the gadgets are constructed recursively using a procedure described below.

Given two gadgets H and H', we can *meld* them as follows. Let $\{x, y\}$ be a replacement of an edge that does not fan out toward the node of H, and let $y \in core(H)$. Select $\{x', y'\}$ similarly. Then $meld(H, H')$ is created by discarding x and x', and by identifying y with y'. Fig. 3 shows several 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.

Lemma 5 *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 $meld(H, H')$. We will prove only (i), the other cases being similar. Property (1) of the gadget correctness is inherited from H and H', because edges that are fanning out in H_0 were fanning out in H or H' and they are represented as before. One can also see that $|rind(H_0)| = i + j - 2$, so H_0 represents a node of degree $i + j - 2$. Property (2) follows quickly from the fact that $|core(H_0)| = |core(H)| + |core(H')| - 1$. To prove property (3), consider U_0 such that $core(H_0) \subset U_0 \subset H_0$, such that $|U_0|$ is even. Let $U = U_0 \cap rind(H)$ and $U' = U_0 \cap rind(H')$. Because $|core(H_0)|$ is odd, $|U| + |U'|$ is also odd. Without loss of generality assume that $|H|$ is even and $|H'|$ is odd. Because H is a correct Si gadget, the subgraph of $core(H) \cup U$ contains a perfect matching. Now it remains to find the matching for $core(H') \cup U'$. We first obtain a matching for $core(H') \cup U' \cup \{x'\}$. We then remove the edge $\{x', y'\}$, because in H' node x' has degree 1, hence this edge must belong to our matching. Note that x' was discarded during melding, and $y' = y$ is already matched, so we have matched all the nodes of U_0. ❑

The following theorem estimates the quality of our gadget reduction of the T-join Problem to the Minimum Cost Perfect matching.

Theorem 3 *Consider an instance of Minimum Cost T-join problem with n nodes, m edges and n_0 nodes of T that have degree 3. In linear time we can generate a strongly equivalent instance of the Minimum Cost Perfect matching that has at most $2m$ nodes and at most $6m - 5n + 0.5n_0$ edges.*

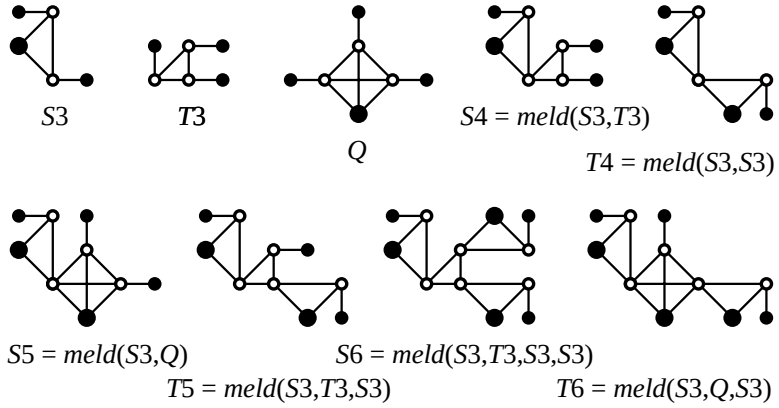

Fig. 3. Replacing a vertex v with G_v. Empty circles indicate the nodes of G_u, solid circles indicate nodes that are connected to G_u via replacement edges. A large solid circle indicates an edge that fans out toward v.

Proof. First, we reduce the problem to the case of a biconnected graph, thus eliminating the nodes of degree 0 and 1. Second, we decide for each edge the direction in which it can be fanned out. In the third stage, we replace each node v with its respective gadget H_v. Fourth, and finally, we connect the gadgets, making sure that if H_u assumes that the edge $\{u, v\}$ fans out toward u, we allowed for that in the second stage. Fig. 2 illustrates the last three stages of this process.

It is easy to see that a node of degree d is replaced with a gadget with d or $d-1$ nodes, thus the total number of nodes in the new graph is bounded by the sum of node degrees in the original graph, i.e., $2m$. The estimate of the resulting number of edges is less straightforward. We consider two classes of nodes: "original", and "extras" – extra replacements and connections. Obviously, we have exactly m original edges. For the extras, one can check that for a node v of degree $d \leq 5$, H_v contains at most $2.5d - 5$ extra edges, with the exception of T-nodes of degree 3, that have $3 = 2.5 \times 3 - 5 + 0.5$ extra edges. Moreover, melding with Q increases d by 2, and the number of extra edges by 5. By adding these expressions for all nodes we obtain the claimed inequality. ❑

Obviously, the smaller the graph that is produced by our transformation, the less time we will need to run an algorithm for Minimum Cost Perfect matching. Can one show a set of provably minimal gadgets? We can answer this question partially, i.e., the number of nodes cannot be decreased in any of our gadgets. We also conjecture that our gadgets use the minimum number of extra edges as well. For degrees smaller than 6 we have verified this conjecture by an exhaustive case analysis.

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

Theorem 4 *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 of Ackerman function.*

4 Computational Experience

For the VLSI mask layout application, we have implemented several approaches, including the reduction to perfect matching using the gadgets we have described, in C++ on a Unix platform. For solving the Perfect Matching problem, we have used the most recent and fastest implementation, due to Cook and Rohe [5]. 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 design, as well as other sparse instances of the T-join problem. Table 1 also confirms that our new exact method significantly improves over the previous methods of [15] [10]: it reduces by 40% the number of unresolved phase 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 algorithm is around 10%, which is significantly larger than the 2% for Euclidean matchings reported in [18].

References

1. A. V. Aho, J. E. Hopcroft and J. D. Ulman, *The Design and Analysis of Computer Algorithms*, Addison Wesley, Reading, MA, 1974.
2. F. Barahona, "Planar multicommodity flows, max cut and the Chinese postman problem ", In W. Cook and P. D. Seymour, eds., Polyhedral Combinatorics, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1 (1990), pp. 189-202.
3. 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", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, 1999, to appear.
4. W. J. Cook, W. H. Cunningham, W. R. Pulleyblank and A. Shrijver, *Combinatorial Optimization*, Willey Inter-Science, New York, 1998.
5. W. Cook and A. Rohe, "Computing Minimum-Weight Perfect Matchings", *http://www.or.uni-bonn.de/home/rohe/matching.html*, manuscript, August, 1998.
6. H. N. Gabow and R. E. Tarjan, "Faster scaling algorithms for general graph matching problems", Journal of the ACM 38 (1991) 815-853.
7. M. X. Goemans and D. P. Williamson, "A general approximation technique for constrained forest problems", SIAM Journal on Computing 24 (1995) 296-317.

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

Algorithms	#conflicts	runtime	#conflicts	runtime	#conflicts	runtime
Greedy[15]	2650	.56	2722	3.66	6168	5.38
Voronoi[10]	2340	2.20	2064	4.69	5050	11.07
Iterated Voronoi[3]	1828	2.35	1552	5.46	3494	13.51
GW[7]	1612	3.33	1488	5.77	3280	14.47
Exact (this paper)	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 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 alignment algorithms suggested in [15], [10], [3], 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 128MB RAM.

8. F. O. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time", *SIAM J. Computing* 4(3) (1975), pp. 221-225.
9. A. B. Kahng and H. Wang, "Toward Lithography-Aware Layout: Preliminary Litho Notes", manuscript, July 1997.
10. 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.
11. 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.
12. R.J. Lipton and R.E. Tarjan, "A separator theorem for planar graphs", *SIAM J. Appl. Math.*, 36 (1979), pp. 177-189.
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. M. Müller-Hannemann and K. Weihe, "Improved Approximations for Minimum Cardinality Quadrangulations of Finite Element Meshes", *Proc. ESA'97, Graz, Austria*, pp. 364-377.
15. 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.
16. G. I. Orlova and Y. G. Dorfman, "Finding the Maximum Cut in a Graph", *Engr. Cybernetics* 10 (1972), pp. 502-506.
17. SIA, *The National Technology Roadmap for Semiconductors*, Semiconductor Industry Association, December 1997.
18. D. P. Williamson and M. X. Goemans, "Computational experience with an approximation algorithm on large-scale Euclidean matching instances", INFORMS Journal of Computing, 8 (1996) 29-40.