

Approximating the Maximum Sharing Problem

Amitabh Chaudhary^{1*}, Danny Z. Chen^{1**}, Rudolf Fleischer^{2***}, Xiaobo S. Hu¹, Jian Li², Michael T. Niemier^{1,3†}, Zhiyi Xie², and Hong Zhu^{2‡}

¹ Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: {achaudha, dchen, shu, mniemier}@cse.nd.edu

² Department of Computer Science and Engineering, Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai 200433, China. E-mail: {lijian83, rudolf, xie_zhiyi, hzhu}@fudan.edu.cn

³ College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA. E-mail: mniemier@cc.gatech.edu

Abstract. In the *maximum sharing problem (MS)*, we want to compute a set of (non-simple) paths in an undirected bipartite graph covering as many nodes as possible of the first node layer of the graph, with the constraint that all paths have both endpoints in the second node layer and no node in that layer is covered more than once. MS is equivalent to the *node-duplication based crossing elimination problem (NDCE)* that arises in the design of molecular quantum-dot cellular automata (QCA) circuits and the physical synthesis of BDD based regular circuit structures in VLSI design. We show that MS is NP-hard, present a polynomial-time 1.5-approximation algorithm, and show that MS cannot be approximated with a factor better than $\frac{740}{739}$ unless $P = NP$.

1 Introduction

Let $G = (U, V; E)$ be an undirected bipartite graph in which U is the *upper* node layer and V is the *lower* node layer. Let $m = |E|$ and $n = |U| + |V|$. In the *maximum sharing (MS) problem* we want to find a set of (non-simple) paths

* The order of authors follows the international standard of alphabetic order of the last name. In China, where first-authorship is a particularly important aspect of a publication, the order of authors should be J. Li, A. Chaudhary, D.Z. Chen, R. Fleischer, X.S. Hu, M.T. Niemier, Z. Xie, and H. Zhu.

** The research of this author was supported in part by the US National Science Foundation under Grant CCF-0515203. This work was partially done while the author was visiting and supported in part by a grant from the Shanghai Key Laboratory of Intelligent Information Processing at Fudan University, China.

*** We gratefully acknowledge support from the National Natural Science Fund China (grant no. 60573025).

† The research of this author was supported in part by the US National Science Foundation under Grant CCF-0404011.

‡ This work is supported by the China National Natural Science Fund (grant no. 60496321).

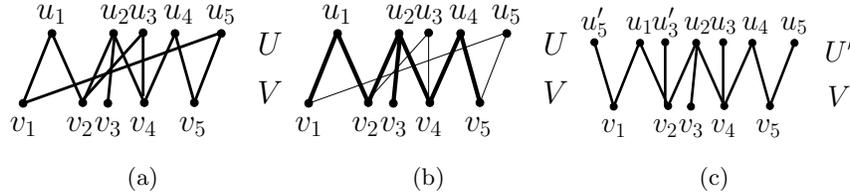


Fig. 1. MS and NDCE: (a) A layout of a two-layer graph G with ten edge crossings; (b) an MS solution with four sharings; (c) the corresponding NDCE solution with two node duplications.

with both endpoints in V maximizing the total length of the paths. Every edge and every node of V can appear at most once in all the paths, except for edges to nodes in V of degree one which may appear twice consecutively in the same path. Note that a node in U can occur multiple times in the same path, and in multiple paths as well. See Fig. 1(b) for an example. Intuitively, a path in an MS solution can be viewed as a concatenated sequence of sharings. A *sharing* is a path (x, u, y) with $u \in U$ and $x, y \in V$, $x \neq y$.

MS has important applications in circuit design. Consider, for example, the (two-layer) *crossing minimization problem* (CM). The two layers of G are laid out on parallel straight lines and each edge is drawn as a straight line segment between an upper node and a lower node. The objective is to minimize the number of edge crossings by reordering the nodes in the two layers. In some applications it is not enough to minimize the number of edge crossings — we must have no crossings at all. This can be achieved by duplicating nodes of the upper layer [3, 4]. When we *duplicate* a node u we create a new node u' and partition the edges incident to u into a set of edges that remain incident to u and a set of edges that become incident to u' . A node may be duplicated multiple times; at each instance some of the currently incident edges are transferred to the new copy. In the *node-duplication based crossing elimination problem* (NDCE) we want to minimize the number of upper nodes to be duplicated such that, after a suitable reordering of the nodes, all edges can be drawn crossing-free as straight lines. See Fig. 1(c) for an example.

An MS solution with s sharings corresponds to an NDCE solution with $m - n - s$ node duplications [4]. Thus, minimizing node duplications in NDCE is equivalent to maximizing sharings in MS.

Good approximations for MS do not necessarily translate into good approximations for NDCE (just consider the case when NDCE has a solution with zero duplications), but a good MS-approximation can serve as a good heuristic for solving NDCE. For example, if the number of sharings is at most a fraction $k/(k+1)$ of the number of upper vertices, for some $k \geq 1$, then NDCE can be approximated to a $(1 + k/3)$ factor in polynomial time, using our 1.5-approximation for MS.

Our results. MS generalizes the NP-hard *maximum weight node-disjoint path cover problem* (MWPC), where we want to find a set of node-disjoint paths in an undirected graph maximizing the number (or total weight) of the edges used by the paths. MWPC is equivalent to MS when all nodes in U have degree two (V and U correspond to the nodes and edges of the MWPC instance, respectively). Thus, MS is also NP-hard.

MWPC is also equivalent to the $(1, 2)$ -TSP problem in the following sense. An approximation ratio of γ for one problem yields an approximation ratio of $\frac{1}{2-\gamma}$ for the other [15] (note that we adapted their formula for the approximation ratio to our different definition of approximation ratio). Since $(1, 2)$ -TSP can be approximated with a factor of $\frac{8}{7}$ [2], MWPC, and thus the case of MS where all nodes in U have degree two can be approximated with a factor of $\frac{7}{6}$. On the other hand, it is NP-hard to approximate $(1, 2)$ -TSP better than with a factor of $\frac{741}{740}$ [10]. Thus, MS cannot be approximated with a factor better than $\frac{740}{739}$ unless $P = NP$. This lower bound could also be derived directly from an approximation preserving reduction from $(1, 2)$ -TSP to MWPC.

The main result of this paper is a polynomial-time 1.5-approximation algorithm for MS by a reduction to the *color path packing problem* (CPP). A maximum matching in the graph of the CPP instance would give us a 2-approximation for MS. We can solve a relaxation of CPP, the *color path-cycle packing problem* (CPCP), optimally in polynomial time by computing a maximum matching in a related graph. In a non-trivial step we can then transform an optimal CPCP solution back to a 1.5-approximation solution for MS. The 2-matching algorithm would also give us a 1.5-approximation for MWPC.

Related work. CM has been studied in the context of graph drawing [7], visualization [6], DNA mapping [17], and optimization of circuit layouts in terms of wiring congestion, total wire length, and layout area (e.g., see [3, 14]). It is NP-hard [13], even when one layer of nodes is already in a fixed order [8] (so-called *fixed-layer* CM). No approximability results are known for CM.

A related problem is to determine whether a given bipartite graph has a (not necessarily induced) planar subgraph with at least k edges, for a given k . This problem, too, was shown to be NP-complete by Eades and Whitesides [7]. It remains NP-complete even for the fixed-layer case. If both layers have a fixed ordering, then there is a polynomial-time algorithm. Another related problem is when only a given set of edge crossings is considered as restricted, and the objective is to minimize the restricted crossings; Finocchi [11] gave a 2-approximation solution for this problem.

NDCE was introduced (and solved by an integer linear program formulation) by Chaudhary et al. [4] to solve layout problems in the design of molecular quantum-dot cellular automata (QCA) circuits [1, 16]. QCA circuits are currently the focus of increasingly intense research efforts aimed at building logic gates at the nanoscale. A major obstacle to building QCA circuits is that chemists are finding that it is considerably difficult to fabricate wire crossings in molecular QCA circuit layouts. Thus, some of the current research efforts are focused on

building QCA circuits with no crossings in their layouts. Hence the need for solving NDCE.

Another application for crossing elimination can be found in the physical synthesis of Binary Decision Diagram (BDD) based regular circuit structures [3]. In contrast to CM, fixed-layer NDCE (where the order of the nodes in V is fixed) can be solved in linear time [4]. For NDCE on general non-bipartite graphs, a heuristic method was proposed by Cao and Koh [3] but they did not give any guarantees on the quality of the solution.

Sharings were introduced by Chaudhary et al. [4]. In an earlier paper [5], the authors studied the *maximum simple sharing problem (MSS)*, where also the upper nodes can only be visited at most once by the paths. By relaxing the path constraint to also allow cycles, they were able to obtain a $\frac{5}{3}$ -approximation for MSS. Although the two problems seem to be closely related, the techniques used for solving MSS are not helpful for solving MS.

2 Approximating MS: A Special Case

In this section we present a polynomial-time 1.5-approximation algorithm for MS under the assumption that all lower nodes in V have degree at least two. In the next section we show how to extend it to also handle degree-one nodes.

Let $G = (U, V; E)$ be an undirected bipartite graph such that the nodes in V have degree at least two. We will transform the MS problem on G into a color path-cycle packing problem (CPCP) on a graph G_V generated from G . CPCP captures the key structure of MS, and we can solve it optimally in polynomial time by reduction to a maximum matching problem. From an optimal CPCP solution on G_V we can then obtain a set of feasible paths forming a 1.5-approximate solution for MS on G .

2.1 The Color Path-Cycle Packing Problem (CPCP)

Consider the following undirected graph $G_V = (V, E_V)$, which intuitively collapses every sharing in G into a single edge between two lower nodes. The node set of G_V is just the lower node set V . For any two distinct nodes v, w in V , if v and w can form a sharing in G , then we put the edge (v, w) in E_V . We associate with each edge $e = (v, w)$ in E_V a set C_e of upper nodes of G , called *colors*, such that if v and w can form a sharing in G through an upper node u , then u is included as a color in C_e .

MS is equivalent to the *color path packing problem (CPP)* on G_V , where we want to find a subgraph H of G_V with the maximum number of edges consisting of a set of node-disjoint simple paths in G_V such that we can color each edge e of H by a color in C_e without coloring any two consecutive edges in any path of H by the same color. Since MS is NP-hard, CPP is also NP-hard.

To approximate MS, we actually need to relax CPP by allowing simple cycles as well as simple paths in the sought subgraph of G_V . We also relax the color constraints. We call this relaxed version the *color path-cycle packing problem*

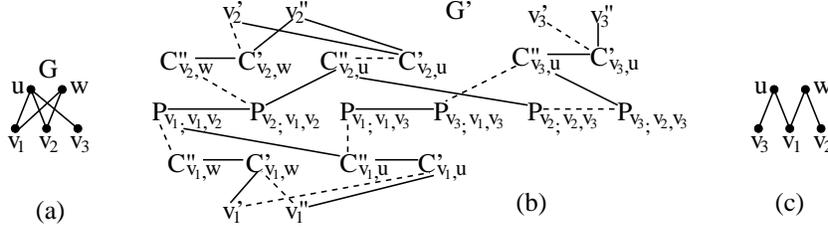


Fig. 2. Transforming an MS problem on G into a maximum matching problem on G' . (a) A simple bipartite graph G ; (b) the corresponding graph G' with a maximum matching (the dashed edges); (c) the corresponding sharing path (v_3, u, v_1, w, v_2) in G .

(CPCP). Formally, CPCP is the problem to find a subgraph H of G_V with the maximum number of edges such that every node in H has degree at most two and we can color both endnodes of every edge e of H by a color in C_e such that the two colors corresponding to the two incident edges of a node are different. Note that an edge may assign different colors to its two endpoints. In other words, an edge may or may not change its color halfway between the endpoints.

Clearly, the optimal objective function value of CPCP is at least as large as that of CPP. We show below that we can obtain a CPP solution SOL_{CPP} from any CPCP solution SOL_{CPCP} such that $|SOL_{CPP}| \geq \frac{2}{3}|SOL_{CPCP}|$. Since we can solve CPCP optimally in polynomial time by reduction to a maximum matching problem, we obtain a $\frac{2}{3}$ -approximation for CPP.

2.2 Solving CPCP

We construct an undirected graph $G' = (V', E')$ from G and G_V as follows (see Fig. 2). For every node $v \in V$ we put two nodes v' and v'' in V' , called V -type nodes. For every edge (v, u) in E with $v \in V$ and $u \in U$ we add to V' two nodes $C'_{v,u}$ and $C''_{v,u}$, called C -type nodes, and to E' three edges $(v', C'_{v,u})$, $(v'', C'_{v,u})$, and $(C'_{v,u}, C''_{v,u})$. For any two nodes v_1, v_2 in G that can form a sharing through an upper node $u \in U$ we add to V' two nodes $P_{v_1;v_1,v_2}$ and $P_{v_2;v_1,v_2}$, called P -type nodes, and to E' an edge $(P_{v_1;v_1,v_2}, P_{v_2;v_1,v_2})$; moreover, we add edges $(P_{v_1;v_1,v_2}, C''_{v_1,u})$ and $(P_{v_2;v_1,v_2}, C''_{v_2,u})$ to E' . These are the nodes and edges of G' . Note that V' contains exactly $2 \cdot |V|$ V -type nodes, $2 \cdot |E|$ C -type nodes, and $2 \cdot |E_V|$ P -type nodes.

This construction transforms the MS problem on G to a maximum matching problem on G' (by relaxing some constraints of the MS problem). Fig. 2(c) gives an example showing a sharing path in G corresponding to a matching in G' .

Theorem 1. *Suppose G_V has an optimal CPCP solution SOL whose value is $|SOL|$, and G' has a maximum matching M . Then, $|M| = |E_V| + |E| + |SOL|$.*

Proof. First we prove $|M| \geq |E_V| + |E| + |SOL|$. Given an optimal CPCP solution SOL on G_V , we construct a matching M' of size $|E_V| + |E| + |SOL|$ in G' as

follows. For every edge $e = (v, w) \in SOL$ in G_V whose endpoints are colored by $c_{e,v}$ and $c_{e,w}$ (possibly $c_{e,v} = c_{e,w}$), we add the edges $(v', C'_{v,c_{e,v}}), (C''_{v,c_{e,v}}, P_{v;v,w}), (w', C'_{w,c_{e,w}}),$ and $(C''_{w,c_{e,w}}, P_{w;v,w})$ of G' to M' . Note that v' (or w') should be changed to v'' (or w'') if v' (or w') is already matched by an edge of M' . Clearly, these edges are part of a matching in G' .

After adding these edges to M' , each unsaturated pair of P -type nodes $P_{v_1;v_1,v_2}$ and $P_{v_2;v_1,v_2}$ can be matched by adding the edge $(P_{v_1;v_1,v_2}, P_{v_2;v_1,v_2})$ to M' . Each pair of unsaturated nodes $C'_{v,u}$ and $C''_{v,u}$ can be matched by adding the edge $(C'_{v,u}, C''_{v,u})$ to M' . Note that for every edge in SOL , our construction of M' saturates two V -type nodes in G' ; further, all C -type nodes and P -type nodes are saturated. Therefore, the number of saturated nodes in G' is exactly $2|SOL| + 2|E| + 2|E_V|$, and $|M'| = |SOL| + |E| + |E_V|$.

Now we prove $|M| \leq |E_V| + |E| + |SOL|$. Given a maximum matching M in G' , it is sufficient to construct a CPCP solution SOL' of size $|SOL'| = |M| - |E_V| - |E|$ in G_V . Note that there exists a maximum matching in G' such that all the P -type nodes and C -type nodes are saturated (the set of all edges of the types of $(C'_{v,u}, C''_{v,u})$ and $(P_{v_1;v_1,v_2}, P_{v_2;v_1,v_2})$ forms a matching in G' saturating all P -type and C -type nodes; then start Edmonds' algorithm [9]). Suppose w.l.o.g. that M has this property. Let n_s denote the number of saturated V -type nodes in M .

Note that if any two nodes $P_{v_1;v_1,v_2}$ and $P_{v_2;v_1,v_2}$ are not matched by the edge $(P_{v_1;v_1,v_2}, P_{v_2;v_1,v_2})$ in M , then both nodes must each be matched with some C -type nodes. Let the corresponding edges in M be $(P_{v_1;v_1,v_2}, C''_{v_1,c_1})$ and $(P_{v_2;v_1,v_2}, C''_{v_2,c_2})$. Then, SOL' contains the edge $e = (v_1, v_2)$, and the colors assigned to (v_1, v_2) in SOL' are $c_{e,v_1} = c_1$ and $c_{e,v_2} = c_2$.

We first argue that $n_s = 2 \cdot |SOL'|$. Since all P -type and all C -type nodes are saturated, for each pair of P -type nodes, say $P_{v_1;v_1,v_2}$ and $P_{v_2;v_1,v_2}$, that are not matched by the edge $(P_{v_1;v_1,v_2}, P_{v_2;v_1,v_2})$ in M , there is a one-to-one correspondence with a pair of saturated V -type nodes: one of v'_1 or v''_1 and one of v'_2 or v''_2 .

Next, we prove that SOL' is indeed a CPCP solution on G_V . It is easy to see that the degree of every node v in SOL' is at most two, because each of $v', v'' \in V'$ can contribute to at most one edge adjacent to v in SOL' . Suppose $e_1 = (u, v), e_2 = (v, w) \in SOL'$ are two adjacent edges, which means that in G' , $P_{v;u,v}$ is not matched with $(P_{v;u,v}, P_{u;u,v})$ and $P_{v;v,w}$ is not matched with $(P_{v;v,w}, P_{w;v,w})$. W.l.o.g. assume that $P_{v;u,v}$ is matched with $(P_{v;u,v}, C''_{v,c_1})$ and $P_{v;v,w}$ is matched with $(P_{v;v,w}, C''_{v,c_2})$. Then we have $c_1 \neq c_2$, and the label colors $c_{e_1,v} = c_1 \neq c_2 = c_{e_2,v}$ (otherwise, $C''_{v,c_1} = C''_{v,c_2}$ would be adjacent to two different edges in the matching M , a contradiction). Therefore, SOL' is a feasible CPCP solution. Since $2|M| = n_s + 2|E_V| + 2|E|$ and $n_s = 2|SOL'|$, we conclude $|SOL'| = |M| - |E_V| - |E|$. \square

Corollary 1. *There is a polynomial-time algorithm for computing an optimal CPCP solution on G_V .*

Proof. The maximum matching problem on G' can be solved in $O(\sqrt{|V'|} \cdot |E'|)$ time [12]. The proof of Theorem 1 shows how to obtain in polynomial time an optimal CPCP solution SOL' in G_V from a maximum matching M in G' . \square

2.3 A 1.5-Approximation for CPP and MS

In this subsection, we show how to obtain a 1.5-approximate CPP solution SOL on G_V from an optimal CPCP solution SOL' on G_V . This immediately gives a 1.5-approximation for MS on G . First, we illustrate on an example why a CPCP solution SOL' may fail to be a feasible CPP solution, and how we can make it feasible by removing some edges from SOL' . Let $P = (v_1, v_2, v_3, v_4)$ be a path in G_V with $C_{(v_1, v_2)} = \{c_1\}$, $C_{(v_2, v_3)} = \{c_1, c_2\}$, and $C_{(v_3, v_4)} = \{c_2\}$. A CPCP solution could contain all three edges with the following color labeling: $c_{(v_1, v_2), v_1} = c_{(v_1, v_2), v_2} = c_1$, $c_{(v_2, v_3), v_2} = c_2$, $c_{(v_2, v_3), v_3} = c_1$, and $c_{(v_3, v_4), v_3} = c_{(v_3, v_4), v_4} = c_2$. This is not a feasible CPP solution. But we can remove the middle edge (v_2, v_3) to make it feasible.

Below we show how we can remove some edges from SOL' to obtain a CPP solution. Remember that we want to remove at most one third of all edges from SOL' .

Let $P = (v_1, \dots, v_\ell)$ be a path in SOL' with label colors such that $c_{(v_{i-1}, v_i), v_i} \neq c_{(v_i, v_{i+1}), v_i}$ for any i with $1 < i < \ell$. We say P can be *feasibly colored* if the coloring of P can be converted to a feasible CPP coloring by carefully choosing color labels for its *edges* (not for its *nodes*) from the color labels of its nodes. To be more precise, we can label each edge (v_{i-1}, v_i) of P by a color $c_{(v_{i-1}, v_i)} \in \{c_{(v_{i-1}, v_i), v_{i-1}}, c_{(v_{i-1}, v_i), v_i}\}$ for $1 < i \leq \ell$, such that $c_{(v_{i-1}, v_i)} \neq c_{(v_i, v_{i+1})}$, for $1 < i < \ell$.

Lemma 1. *Paths in SOL' of length at most 2 are feasibly colorable.* □

Note that in G two different nodes $v_1, v_2 \in V$ cannot simultaneously have sharings with two distinct upper nodes $u, w \in U$ in a feasible MS solution. Such short cycles cannot occur in SOL' (for example, in Fig. 2(a), $v_1, v_2 \in V$ cannot simultaneously have sharings with $u, w \in U$). It might appear that such infeasible simultaneous sharings might correspond to a “degenerate” cycle (v_1, v_2, v_1) in SOL' . The next lemma shows that there is no such “degenerate” cycle in SOL' .

Lemma 2. *Every cycle in SOL' has length at least three.* □

SOL' may contain simple paths and simple cycles. We first deal with the paths in SOL' . Let $P = (v_0, \dots, v_{t-1})$ be a path in SOL' . We remove the edges (v_{3k-1}, v_{3k}) from P , for $k = 1, \dots, \lfloor \frac{t}{3} \rfloor$. The remaining parts of P are a set of paths of length at most two. By Lemma 1, these paths can be feasibly colored. Note that we deleted no more than one third of the edges of SOL' .

Handling cycles in SOL' is more complicated. By Lemma 2, the length of each cycle in SOL' is at least three. We distinguish three cases based on the cycle length. An edge $e = (u, v)$ in SOL' with label colors $c_{e,u} = c_{e,v}$ is called a *1-color edge*, otherwise a *2-color edge*.

Lemma 3. (a) *A simple path consisting of successive 1-color edges and at most two 2-color edges, one at each end of the path, that does not form a cycle can be feasibly colored.*

(b) *A simple path consisting of successive 2-color edges and at most one 1-color edge at one end of the path that does not form a cycle can be feasibly colored.*

Proof. We only show part (b). Consider a simple path $P = (v_1, \dots, v_\ell)$, in which (v_1, v_2) is a 1-color edge and the other edges are 2-color edges. First, we label (v_1, v_2) with color $c_{(v_1, v_2), v_1}$. Since $c_{(v_2, v_3), v_2} \neq c_{(v_2, v_3), v_3}$, at least one of them is not equal to $c_{(v_1, v_2), v_1}$. Thus, we can label (v_2, v_3) with this color. Similarly, we can feasibly color the other 2-color edges of P . \square

Lemma 4. *For any cycle $C \in SOL'$ of length at least four, there exists a subpath of length three in C that can be feasibly colored.*

Proof. If C contains only 2-color edges or only 1-color edges (but not both types), then, by Lemma 3, any three consecutive edges of C can be feasibly colored. If C includes edges of both types, then consider a maximal subpath P of C consisting of only 1-color edges. P is certainly not a cycle. If $|P| \geq 3$, then the lemma holds for P . If $|P|$ is 1 (or 2), then we take two (or one) of the edges adjacent to the endnodes of P (P together with these edges does not form a cycle, because $|C| > 3$). The subpath of C formed by P and these edges can be feasibly colored by Lemma 3(a). \square

Lemma 5. *For any cycle $C \in SOL'$ of length at least five in which the 1-color edges and 2-color edges do not appear alternatingly, there exists a subpath of length four in C that can be feasibly colored.* \square

Case 1: $C = (v_0, v_1, \dots, v_{3t-1}, v_0)$ is a cycle of length $3t$. We remove the edges (v_{3k}, v_{3k+1}) from C , for $k = 0, \dots, t-1$. The remaining parts of C are a set of paths of length exactly two which can be feasibly colored by Lemma 1.

Case 2: C is a cycle of length $3t+1$. We want to remove t edges, resulting in one path of length three and $t-1$ paths of length two which can all be feasibly colored. By Lemma 4, we can find three successive edges of C that can be feasibly colored (and thus be kept in the CPP solution SOL). Next, we remove the two edges of C adjacent to this length-three subpath (if $t = 1$, then there is only one such adjacent edge). If $t \geq 2$, what is left from C at this point is a path P of length $3t-4$. By using the same scheme as for handling the path case, we remove $t-2$ edges from P and obtain a set of paths of length at most two. Note that we remove a total of t edges from C , which is less than a third of all edges.

Case 3: C is a cycle of length $3t+2$. Similarly to Case 2, if we can find four successive edges in C that can be colored feasibly by Lemma 5, then we are done. Suppose we cannot; then by Lemma 5, the 1-color edges and 2-color edges in C must appear alternatingly, and thus t must be an even integer. Let $C = (v_0, \dots, v_{t'-1})$, where $t' = 3t+2$, be a cycle in which the 1-color edges and 2-color edges appear alternatingly and (v_0, v_1) is a 1-color edge. We remove the edges (v_{4k}, v_{4k+1}) , for $k = 0, \dots, \lfloor \frac{t'-1}{4} \rfloor$. The remaining parts of C are all paths of length at most three which can be feasibly colored by Lemma 3(a). Hence, in this case we remove a total of $\lfloor \frac{t'-1}{4} \rfloor + 1 \leq t$ edges, which is less than a third of all edges.

Theorem 2. *MS can be approximated within a factor of 1.5 in polynomial time if there are no degree-one lower nodes.*

Proof. The claim follows from Corollary 1. The running time of our MS approximation algorithm is dominated by the step of computing a maximum matching in the graph G' , whose numbers of nodes and edges are a low degree polynomial in the numbers of nodes and edges of the input graph G . \square

2.4 A More Practical 1.5-Approximation Algorithm

The technique described in the previous subsection gives the currently best approximation ratio for MS. In practice, however, we can do better, since many edges removed are unnecessary. Here, we propose a more practical algorithm to compute a CPP solution by removing edges from a CPCP solution. Note that, in the worst case, the scheme in this subsection also gives a 1.5-approximation.

Given a CPCP solution SOL_{CPCP} that consists of some cycles and paths, we first analyze the structures that make SOL_{CPCP} not a feasible CPP solution. One reason could be the existence of cycles, and we need to remove at least one edge on each cycle.

Another problem might be a subpath formed by 2-color edges with two 1-color edges at each end. For example, consider a simple path $P = (v_1, \dots, v_\ell)$ in which ℓ is even, (v_1, v_2) and $(v_{\ell-1}, v_\ell)$ are both 1-color edges, and the other edges are 2-color edges. Moreover, colors are assigned as follows: $c_{(v_1, v_2), v_1} = c_{(v_2, v_3), v_3} = \dots = c_{(v_{\ell-2}, v_{\ell-1}), v_{\ell-1}} = c_1$, and $c_{(v_2, v_3), v_2} = c_{(v_3, v_4), v_4} = \dots = c_{(v_{\ell-2}, v_{\ell-1}), v_{\ell-2}} = c_{(v_{\ell-1}, v_\ell), v_\ell} = c_2$. Clearly, this subpath cannot be feasibly colored without edge removals. To see why this is the only case when we cannot feasibly color all edges, consider an optimal deletion of a 1-color edge (v_1, v_2) . It should be a 1-color edge because we cannot feasibly color (v_1, v_2) with its color $c_{(v_1, v_2), v_1} = c_1$. This is because one of its adjacent 2-color edges, say (v_2, v_3) , would be forced to be colored with $c_1 = c_{(v_2, v_3), v_3}$ (note that $c_{(v_2, v_3), v_2} \neq c_1$). If there is another choice to color (v_2, v_3) (i.e., using color $c_{(v_2, v_3), v_2}$) without changing other edges' colors, we could add (v_1, v_2) to the solution SOL_{CPCP} , contradicting the optimality. But why is (v_2, v_3) forced to be colored with $c_1 = c_{(v_2, v_3), v_3}$? That is because its adjacent 2-color edge (v_3, v_4) is forced to be colored with $c_1 = c_{(v_2, v_3), v_2}$. We can continue this argument until we encounter a 1-color edge $(v_{\ell-1}, v_\ell)$ which must be colored with $c_{(v_{\ell-1}, v_\ell), v_\ell}$.

If a 2-color edge is deleted in an optimal deletion, the argument is similar and proceeds until meeting the 1-color edges at the two ends of the path. If such a structure occurs, then we cannot feasibly color all edges in the structure, and at least one edge must be deleted. At the same time, it is easy to see that it is sufficient to delete only one 1-color edge in such a subpath. Thus, we can w.l.o.g. assume that an optimal method always removes 1-color edges. Moreover, such subpaths are mutually disjoint except that some pairs of them may share a common 1-color edge (at the ends of such a pair of subpaths).

We construct the following graph D for the edge removal. The nodes of D are all 1-color edges of the CPCP solution. Two nodes of D are linked by an edge if the two corresponding 1-color edges are at the two ends of one subpath described above. Thus any edge removal that makes the CPCP solution a feasible

CPP solution corresponds to a vertex cover in D . Since D is a graph of maximum degree two, we can compute a minimum vertex cover in D in polynomial time.

3 Approximating MS: The General Case

In this section we show how to modify the approximation algorithm of Section 2 to accommodate lower nodes of degree one. We will use the fact that we may assume w.l.o.g. that each upper node is connected to at most one degree-one lower node (if we can connect crossing-free to one such node, we can connect crossing-free to many such nodes).

Given G , we construct G_V exactly as before. We must slightly relax the color constraint in the definition of CPP. If a path visits a degree-one node in V , it may have two consecutive occurrences of the same edge leading to that node, which of course must have the same color. Now, MS on G is again equivalent to CPP on G_V . For CPCP, we need a similar relaxation of the color constraint at degree-one nodes of V . The optimal objective value of CPCP on G_V is then at least as large as that of CPP.

Now we construct the graph G' from G . Let V_S be the set of degree-one lower nodes in G . For each $v \in V$ we add two V-type nodes v' and v'' , for a total of $2 \cdot |V|$ such nodes. For each edge (v, u) in G with $v \in V$ and $u \in U$, if $v \notin V_S$, we add two C-type nodes $C'_{v,u}$ and $C''_{v,u}$, and we add edges as described before. If, however, $v \in V_S$, we add four C-type nodes: $C'_{v_a,u}$ and $C''_{v_a,u}$, as well as $C'_{v_b,u}$ and $C''_{v_b,u}$. Further, we add the following six edges: $(v', C'_{v_a,u})$, $(v'', C'_{v_a,u})$, and $(C'_{v_a,u}, C''_{v_a,u})$, and $(v', C'_{v_b,u})$, $(v'', C'_{v_b,u})$, and $(C'_{v_b,u}, C''_{v_b,u})$. Thus the total number of C-type nodes is $2 \cdot (|E| + |V_S|)$. Finally, we add $2 \cdot |E_V|$ P-type nodes exactly as before. The edges between P-type and C-type nodes are added just as before, except when for an edge $(v_1, v_2) \in E_V$ in which one (and there can be only one) of the nodes, say v_2 , is in V_S . In that case, let v_1 and v_2 have a common upper node neighbor u (again, only one common neighbor is possible). We add the following four edges to E' : $(P_{v_1;v_1,v_2}, P_{v_2;v_1,v_2})$, $(P_{v_1;v_1,v_2}, C''_{v_1,u})$, $(P_{v_2;v_1,v_2}, C''_{v_2a,u})$, and $(P_{v_2;v_1,v_2}, C''_{v_2b,u})$.

Theorem 3. *Suppose G_V has an optimal CPCP solution SOL whose value is $|SOL|$, and G' has a maximum matching M . Then, $|M| = |E_V| + |E| + |V_S| + |SOL|$. \square*

Informally, the only change in G' is that there are now more C-type nodes — $2 \cdot |V_S|$ more nodes. See Fig. 3 for an illustration.

The final step is to show that given any CPCP solution on G_V , say SOL' , we can obtain a solution SOL for CPP on G_V such that $|SOL| \geq \frac{2}{3}|SOL'|$. Let $v \in V_S$ be a degree-one lower node that is a neighbor of the upper node $u \in U$. Suppose SOL' consists of a path or cycle with the three nodes $\sigma_v = (v_1, v, v_2)$ in order, and with corresponding colors $c_{(v_1,v),v_1}$, $c_{(v_1,v),v}$, $c_{(v,v_2),v}$, and $c_{(v,v_2),v_2}$. Now $c_{(v_1,v),v} = c_{(v,v_2),v} = u$ (this is a legal coloring). Observe that if there is no such sequence σ_v in SOL' for any $v \in V_S$, then SOL' is also a solution to CPCP

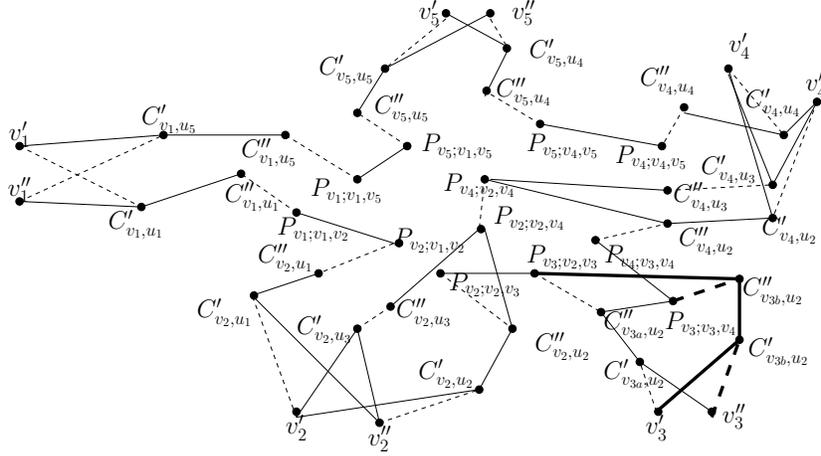


Fig. 3. The graph G' corresponding to G in Fig. 1(a). The thick edges are the extra edges required due to degree-one lower nodes. The dashed edges are the matching corresponding to the CPCP solution.

on G_V , and we can directly use the algorithm in Section 2.3 to obtain a solution SOL for CPP on G_V .

So suppose such a sequence σ_v exists. There can be at most one sequence for each $v \in V_S$. Take the path (or cycle) P' containing v and break it into two paths (or one path, respectively) at the node v . In other words, the sequence (v_1, v) is separated from (v, v_2) . Let the resulting path(s) be denoted by P . Remove the third edge for each path in P , as in Section 2.3, followed by an application of Lemma 1, to obtain paths in P in which exactly one color from C_e is assigned to each edge e , and no two consecutive edges have the same color. P may, however, have two occurrences of v and thus cannot be part of a solution for CPP. We remedy this by combining back the two separate ends (if they both exist) to form the original sequence (v_1, v, v_2) . This will lead to two consecutive edges having the same color, but that is a legal coloring.

Does this recombination create a cycle? Observe that if the original sequence σ_v is part of a cycle, the cycle has at least three edges. This follows from reasoning very similar to the proof of Lemma 2. If P consists of a single path, it has at least three edges, and thus at least one edge is removed. Thus, recombinations do not form a cycle. The process described above is performed on every sequence σ_v , for each $v \in V_S$. For paths or cycles in SOL' not containing such sequences, the process is exactly as described in Section 2.3.

Theorem 4. *MS can be approximated within a factor of 1.5 in polynomial time.* \square

References

1. D.A. Antonelli, D.Z. Chen, T.J. Dysart, X.S. Hu, A.B. Khang, P.M. Kogge, R.C. Murphy, and M.T. Niemier. Quantum-dot cellular automata (QCA) circuit partitioning: problem modeling and solutions. *Proc. 41st ACM/IEEE Design Automation Conf. (DAC'04)*, pages 363–368, 2004.
2. P. Berman and M. Karpinski. $\frac{8}{7}$ -approximation algorithm for (1,2)-TSP. *Proc. 17th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'06)*, pages 641–648, 2006.
3. A. Cao and C.-K. Koh. Non-crossing ordered BDD for physical synthesis of regular circuit structure. *Proc. International Workshop on Logic and Synthesis*, pages 200–206, 2003.
4. A. Chaudhary, D.Z. Chen, X.S. Hu, M.T. Niemier, R. Ravinchandran, and K.M. Whitton. Eliminating wire crossings for molecular quantum-dot cellular automata implementation. *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pages 565–571, 2005.
5. D.Z. Chen, R. Fleischer, J. Li, Z. Xie, and H. Zhu. On approximating the maximum simple sharing problem. *Proc. of the 17th International Symposium on Algorithms and Computation (ISAAC'06)*. Springer LNCS 4288, pages 547–556, 2006.
6. G. Di Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Englewood Cliffs, NJ, 1998.
7. P. Eades and S. Whitesides. Drawing graphs in two layers. *Theor. Comput. Sci.*, 131:361–374, 1994.
8. P. Eades and N.C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.
9. J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
10. L. Engebretsen and M. Karpinski. TSP with bounded metrics. *Journal of Computer and System Sciences*, 72(4):509–546, 2006.
11. I. Finocchi. Layered Drawings of Graphs with Crossing Constraints. *Proc. 9th Annual International Computing and Combinatorics Conference*, pages 357–367, 2001.
12. H. N. Gabow. Data Structures for Weighted Matching and Nearest Common Ancestors with Linking. *Proc. 7th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'90)*, pages 434–443, 1990.
13. M.R. Garey and D.S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.
14. T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Wiley, New York, 1990.
15. C.H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.
16. P.D. Tougaw and C.S. Lent. Logical devices implemented using quantum cellular automata. *J. of App. Phys.*, 75:1818, 1994.
17. M.S. Waterman and J.R. Griggs. Interval graphs and maps of DNA. *Bull. Math. Biol.*, 48(2):189–195, 1986.