On Approximating the Maximum Simple Sharing Problem*

Danny Z. Chen[1]**, Rudolf Fleischer[2]***, Jian Li[2], Zhiyi Xie[2], and Hong Zhu[2]†

[1] Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: dchen@cse.nd.edu

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

Abstract. In the *maximum simple sharing problem (MSS)*, we want to compute a set of node-disjoint simple paths in an undirected bipartite graph covering as many nodes as possible of one layer of the graph, with the constraint that all paths have both endpoints in the other layer. This is a variation of the *maximum sharing problem (MS)* that finds important applications in the design of molecular quantum-dot cellular automata (QCA) circuits and physical synthesis in VLSI. It also generalizes the maximum weight node-disjoint path cover problem. We show that MSS is NP-complete, present a polynomial-time $\frac{5}{3}$-approximation algorithm, and show that it 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 with *upper nodes* U and *lower nodes* V. An upper node $u \in U$ forms a *sharing* with two distinct lower nodes $v_1, v_2 \in V$ if (u, v_1) and (u, v_2) are both edges in E. In the *maximum simple sharing problem (MSS)*, we want to cover the maximum number of upper

* This work was supported in part by a grant from the Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai, China. 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 Zhiyi Xie, Jian Li, Hong Zhu, Danny Z. Chen, and Rudolf Fleischer.

** 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 the Shanghai Key Laboratory of Intelligent Information Processing at Fudan University, China.

*** The work described in this paper was partially supported by a grant from the National Natural Science Fund China (grant no. 60573025).

† The work described in this paper was partially supported by a grant from the National Natural Science Fund China (grants #60496321, #60373021, and #60573025) and the Shanghai Science and Technology Development Fund (grant #03JC14014).

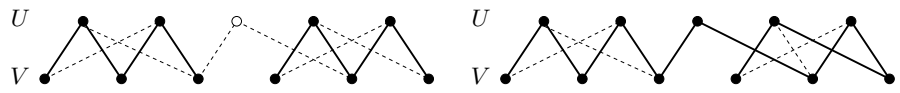

Fig. 1. The MSS problem: A non-optimal maximal solution (left), and an optimal solution (right).

nodes by sharings such that the edges of the sharings form a set of node-disjoint simple paths in G, where every path has both endpoints in V. See Figure 1 for an example.

MSS is a variant of the *maximum sharing problem (MS)* where a node in U may be involved in multiple sharings (i.e., the paths formed by the sharings may be non-simple and may overlap in the nodes of U). In a companion paper [10] we give a 1.5-approximation for MS. Unfortunately, the techniques used for that result do not carry over to the restricted variant MSS.

MS can help us to solve the *node-duplication based crossing elimination problem (NDCE)* [1, 4]. In a two-layered (bipartite) graph, we want to duplicate as few nodes as possible in the first layer such that afterwards all connections between the two layers can be realized crossing-free (after a suitable permutation of the nodes in both layers). Intuitively, each sharing in an MS solution tells us how to avoid a node duplication. MSS is equivalent to *restricted NDCE* where duplicated nodes can only have a single neighbor in V. Variants of NDCE play a key role in the design of molecular quantum-dot cellular automata (QCA) circuits [1, 12, 15] and physical synthesis [3] which is similar to the key role played by the well studied *crossing minimization problem* [9, 11, 14] in the design of traditional VLSI circuit layouts.

MSS also generalizes the NP-hard *maximum weight node-disjoint path cover problem (PC)* where we want to find in an undirected graph a set of node-disjoint paths maximizing the number (or total weight) of the edges used by the paths. It is easy to see that PC is equivalent to MSS when all nodes in U have degree two (V and U correspond to the nodes and edges of the PC instance, respectively). PC is 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 [13] (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], PC, and thus the case of MSS 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 $\frac{741}{740}$ [7]. Thus, MSS cannot be approximated with a factor better than $\frac{740}{739}$ unless $P = NP$.

An MSS solution is *maximal* if it cannot be enlarged by extending any path from its endpoints without destroying the current solution; otherwise, it is *extendible*. Note that we can enlarge any given solution to a maximal solution in polynomial time. The greedy algorithm that always chooses an unused node in V and arbitrarily extends a maximal node-disjoint path in both endpoints

obviously constructs a maximal solution. Since a sharing touches exactly three nodes, each sharing in a maximal solution can only block three sharings in an optimal solution. Thus, any maximal solution is a 3-approximation for MSS.

Our main contribution is to show that MSS can be approximated with a factor of $\frac{5}{3}$. Our algorithm is based on a relaxation of the path constraint to allow solutions to contain node-disjoint simple cycles as well as paths, still maximizing the number of nodes of U covered by the paths and cycles. We call this relaxed version the *cyclic maximum simple sharing problem (CMSS)*. While MSS is NP-hard, CMSS can be solved optimally in polynomial time as a *maximum weight perfect matching problem (MWPM)* [5]. A similar phenomenon occurs in the 2-matching relaxation of TSP [8] which can be computed in polynomial time [6]. The difference is that a 2-matching is a pure cycle cover, whereas CMSS computes a mixture of cycles and paths. Since each cycle in a CMSS solution contains at least two sharings, we can get a 2-approximate MSS solution by removing one sharing from each cycle (thus breaking the cycle into a path). To obtain an approximation factor of $\frac{5}{3}$, we must carefully construct a set of node-disjoint simple paths from an optimal CMSS solution.

We assume in this paper that every node $v \in V$ has degree at least two. In MS we can get rid of degree-one nodes by adding a parallel edge to the one edge connecting the node, giving rise to the so-called q-MS problem in [10]. This approach does not work for MSS. Instead, we must duplicate the upper node adjacent to a degree-one lower node (we can w.l.o.g. assume that there is only one degree-one neighbor) together with all its adjacent edges. Then, maximizing the sharings is still equivalent to minimizing the node duplications.

The rest of this paper is organized as follows. First, we show in Section 2 that MSS is equivalent to restricted NDCE. In Section 3 we show how to solve CMSS in polynomial time. In Section 4 we then show how to transform an optimal CMSS solution into a $\frac{5}{3}$-approximation for MSS.

2 MSS and Restricted NDCE

The input to NDCE is a bipartite graph $G = (U, V; E)$. We want to duplicate as few nodes of U as possible to achieve a crossing-free drawing of G with the nodes U (and their copies) drawn (in some suitable order) along a line (the upper layer) and the nodes V drawn along another parallel line (the lower layer). In restricted NDCE the copies of nodes in U can only have a single neighbor in V. The following theorem shows that minimizing the number of duplications is equivalent to maximizing the number of simple sharings.

Theorem 1. *Given a bipartite graph $G = (U, V; E)$ in which every node has degree at least two, there is a solution of the MSS problem containing m simple sharings if and only if we can duplicate $|E| - |U| - m$ nodes of U to eliminate all wire crossings.*

Proof. Denote the layout of the circuit without wire crossings by $G' = (U', V'; E')$. U' consists of $|U|$ original nodes and the newly duplicated nodes. One way to

achieve crossing-free wires is to duplicate $|E| - |U|$ nodes, i.e., every edge of E has a distinct endpoint in U'. To reduce the number of node duplications we observe that the original nodes (not the duplicated nodes) in U' can connect to more than one node, thus reducing the number of duplications.

Consider a permutation of the nodes in V', and let v_i and v_{i+1} be two consecutive nodes. It is easy to see that v_i and v_{i+1} can have at most one common neighbor in U'; otherwise there would be some wire crossings. It can also be seen that in G' the degree of a node in U' cannot be bigger than two; otherwise edge crossings cannot be avoided because we cannot duplicate nodes in V.

In G, if there are m simple sharings, then we can arrange m pairs of nodes in V consecutively so that each pair of nodes shares a common neighbor in U, thus reducing the duplication number by m. The other direction can be proved by a similar argument. □

3 The Cyclic Maximum Simple Sharing Problem (CMSS)

The *cyclic maximum simple sharing problem (CMSS)* is defined as follows. Given a bipartite graph $G = (U, V; E)$, find a set $\mathcal{C}$ of node-disjoint simple cycles and simple paths in G such that every path begins at a node of V and ends at another node of V, maximizing the number of nodes in U covered by $\mathcal{C}$ (i.e., maximizing the number of sharings). Since any MSS solution is also a CMSS solution, the optimal objective function value of MSS is upper-bounded by the optimal CMSS value.

We now show how to solve CMSS by reducing it to the *maximum weight perfect matching problem (MWPM)* on undirected graphs which can be solved optimally in polynomial time [5]. Given a bipartite graph $G = (U, V; E)$, we construct an undirected graph H as follows. We want to represent every node and every edge of G by a pair of adjacent nodes in H. If a node or an edge in G is not used by any sharing, then the corresponding paired nodes in H are matched by their connecting edge. Otherwise, they are matched by other edges.

Figure 2 shows an example of the construction. For each node $v \in U \cup V$, we add to H two nodes $v^{(1)}$ and $v^{(2)}$ connected by an edge of weight zero. Similarly, for each edge $e \in E$, we add to H two nodes $e^{(1)}$ and $e^{(2)}$ connected by an edge of weight zero. In addition, for each edge $e = (u, v)$, with $u \in U$ and $v \in V$, we add the four edges $(u^{(1)}, e^{(1)})$, $(u^{(2)}, e^{(1)})$, $(v^{(1)}, e^{(2)})$, and $(v^{(2)}, e^{(2)})$, where the first edge has weight one and the other three edges have weight zero. Finally, for any two nodes v_1 and v_2 in V, we add an edge $(v_1^{(2)}, v_2^{(2)})$ of weight zero to H. It is easy to see that we can construct H in time $O(|E| + |V|^2)$.

Theorem 2. *G has a CMSS solution with k sharings if and only if H has a perfect matching of weight k.*

Proof. Figure 2 illustrates the proof. We prove the "only if" direction first. Given a set $\mathcal{C}$ of node-disjoint simple paths and cycles in G with k sharings, we construct a perfect matching M of weight k in H, as follows. We treat $\mathcal{C}$ as a subgraph

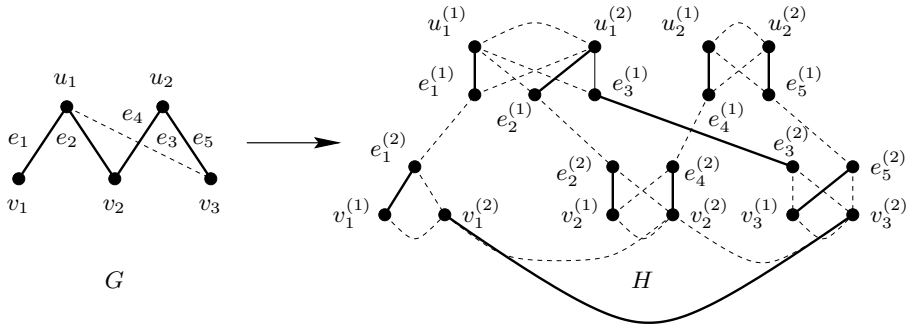

Fig. 2. Illustrating the proof of Theorem 2: The equivalence between CMSS in G and MWPM in H.

of G. For each node v of G not covered by $\mathcal{C}$ we add the edge $(v^{(1)}, v^{(2)})$ to M. Similarly, we add the edge $(e^{(1)}, e^{(2)})$ to M for each edge $e \in E$ not in $\mathcal{C}$.

We now classify the nodes of $\mathcal{C}$ into three types: (i) upper nodes, (ii) lower nodes of degree two in $\mathcal{C}$, and (iii) lower nodes of degree one in $\mathcal{C}$. All nodes of type (i) have degree two in $\mathcal{C}$, and the number of nodes of type (iii) is even.

If $u \in U$ is of type (i), let e_1 and e_2 be the two edges adjacent to u in $\mathcal{C}$. We add the two edges $(u^{(1)}, e_1^{(1)})$ and $(u^{(2)}, e_2^{(1)})$ to M, increasing the weight of M by one. If $v \in V$ is of type (ii), let f_1 and f_2 be the two edges adjacent to v in $\mathcal{C}$. We add the two edges $(v^{(1)}, f_1^{(2)})$ and $(v^{(2)}, f_2^{(2)})$ to M. If $w \in V$ is of type (iii), let g be the edge adjacent to w in $\mathcal{C}$. We add the edge $(w^{(1)}, g^{(2)})$ to M.

All these operations are possible because a node or an edge in G can appear at most once in $\mathcal{C}$. Now, all nodes in H are matched except those of the form $w^{(2)}$ corresponding to a lower node $w \in V$ of type (iii). Because there is an even number of such nodes and they are all pairwise connected, we can arbitrarily match them. Now we have a perfect matching M in H. The weight of M is k, the number of nodes of type (i).

Next, we prove the "if" direction. Given a perfect matching M of weight k in H, we construct a CMSS solution in G, as follows. We call a node $v \in U \cup V$ *used* if the corresponding edge in H, $(v^{(1)}, v^{(2)})$, does not belong to M. Similarly, we call an edge $e \in E$ *used edge* if the corresponding edge in H, $(e^{(1)}, e^{(2)})$, does not belong to M.

Let $e = (u, v)$ be a used edge, where $u \in U$ and $v \in V$. Since $(e^{(1)}, e^{(2)})$ is not in M, $e^{(1)}$ must be matched either with $u^{(1)}$ or $u^{(2)}$, and $e^{(2)}$ must be matched either with $v^{(1)}$ or $v^{(2)}$. Thus, both u and v are used nodes. On the other hand, let $u \in U$ be a used upper node. Since $(u^{(1)}, u^{(2)})$ is not in M, $u^{(1)}$ must be matched with some node $e_1^{(1)}$ and $u^{(2)}$ must be matched with some node $e_2^{(1)}$, where e_1 and e_2 are two edges in E adjacent to u. Thus, both e_1 and e_2 are used edges and there are no other edges of E corresponding to used edges adjacent to u in G. Only the used upper nodes contribute one to the weight of M. Similarly, each used lower node $v \in V$ must be adjacent to one or two used edges in E.

In summary, every used edge connects two used nodes, every used upper node is adjacent to exactly two used edges, and every used lower node is adjacent to either one or two used edges. Thus, all used nodes and used edges form a subgraph $\mathcal{C}$ of G consisting of node-disjoint simple cycles and simple paths such that every path begins at a node of V and ends at another node of V. The number of used nodes in U (i.e., the number of sharings contained in $\mathcal{C}$) equals the weight k of M. □

Corollary 3. *CMSS can be solved optimally in polynomial time.* □

4 Obtaining a $\frac{5}{3}$-approximate MSS Solution

Given a bipartite graph $G = (U, V; E)$, let S denote a (not necessarily optimal) CMSS solution, i.e., S is a subgraph of G. We first classify the lower nodes V into three types: (i) *white nodes*, which are not covered by S, (ii) *gray nodes*, which have degree one in S (i.e., the endpoints of the paths in S), and (iii) *black nodes*, which have degree two in S (i.e., the lower nodes lying in the interior of a path or on a cycle in S). Nodes on a cycle in S are also called *cycle nodes*. Cycle nodes are always black. Note that the color of a lower node depends on the subgraph S and may vary while the subgraph S changes.

Let C be a cycle in S. An edge not belonging to C but connected to an upper node in C is a *short tail* of C. A *long tail* is a chain of two edges not belonging to C starting at a lower node of C with the middle (upper) node not in S (i.e., a long tail is a sharing). We often do not distinguish between short and long tails, just calling them *tails*. A tail of C has the color of its endpoint not in C. Note that the edges of a tail never belong to S. See Figure 3 for an example.

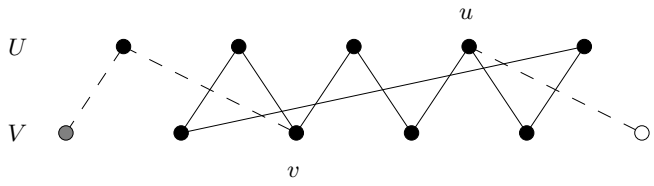

Fig. 3. A gray long tail at v and a white short tail at u.

Lemma 4. *Let C be a cycle in S.*

(a) If C has a white tail, we can break C into a path with the same number of sharings.
(b) If C has a gray tail ending at an endpoint of a path D in S, then we can break C by merging it with D into a single path with the same number of sharings.

(c) If C has a tail ending at a node of another cycle D in S, then we can merge C and D into a single path at the cost of losing one sharing.

Proof. See Fig. 4. □

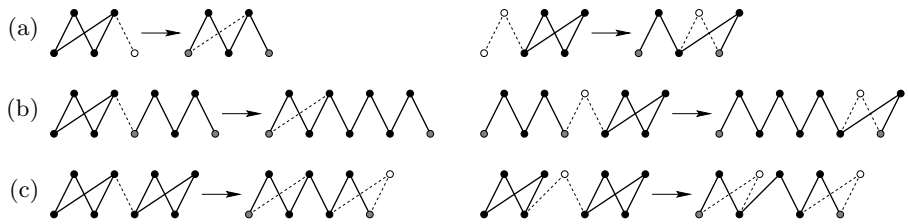

Fig. 4. Three ways of breaking a cycle, proving Lemma 4.

We now present our approximation algorithm. For the input graph $G = (U, V; E)$, let $OPT^\star$ denote an optimal CMSS solution. We apply the following algorithm to $OPT^\star$.

Cycle-breaking Algorithm

Step 0: Let $S = OPT^\star$.

Step 1: Repeatedly pick a cycle C in S with a white or gray tail and break C into a path as in Lemma 4 (a) or (b), until no such cycle exists. Then go to Step 2.

Step 2: Pick a pair of cycles C and D such that C has a tail ending at a cycle node in D, merge C and D into a path P as in Lemma 4 (c), and then go to Step 3. Go to Step 4 if no such pair exists.

Step 3: Let v_1 and v_2 denote the two (gray) endpoints of P. Perform the following two substeps:

- Step 3(a): Repeatedly pick a cycle C that has a tail ending at v_1 or v_2, and break C as in Lemma 4 (b). Then go to Step 3(b).
- Step 3(b): For the two endpoints v_1 and v_2 of P obtained in Step 3(a), if there is an upper node $u \in U$ not in S and both (v_1, u) and (v_2, u) are edges in E, then add the sharing (v_1, u, v_2) to S, closing a cycle. Then go to Step 2.

Step 4: Break all remaining cycles in S by arbitrarily removing one sharing from each cycle.

The output from the algorithm above is our approximate MSS solution. It is easy to see that the algorithm takes only polynomial time. Note that Steps 2 and 3 are iterated at most $\lfloor \frac{|U|}{2} \rfloor$ times since each iteration merges at least two cycles into either one path or one cycle. Moreover, if at the end of an iteration of Steps 2 and 3 a path is generated, then it will stay as a path from that point on.

Let $SOL^\star$ denote the CMSS solution before we begin with Step 4. Furthermore, let $|OPT^\star|$ and $|SOL^\star|$ denote the number of sharings contained in $OPT^\star$ and $SOL^\star$, respectively.

Lemma 5. *Let p be the number of paths that are generated and added to $SOL^\star$ in Steps 2 and 3. Then $|OPT^\star| = |SOL^\star| + p$.*

Proof. Let $s(S)$ denote the number of sharings in S during the algorithm. By Lemma 4 (a) and (b), we do not lose any sharings in Step 1. Thus, $s(S) = |OPT^\star|$ at the end of Step 1. For each iteration of Steps 2 and 3, by Lemma 4 (b) and (c), we lose one sharing in Step 2, but we do not lose any sharing in Step 3(a). Therefore, if the algorithm does not find a suitable upper node in Step 3(b), then one path is generated that will stay as a path from that point on and we lose one sharing. On the other hand, if the algorithm forms a cycle in Step 3(b), then no path is generated and no sharing is lost. Hence, the number of paths generated is equal to the number of sharings lost in Steps 2 and 3, which is exactly p. Therefore, $|OPT^\star| = |SOL^\star| + p$. □

Lemma 6. *Let (v_1, u, v_2) be a sharing in G such that the upper node u is not covered by a path in $SOL^\star$.*

(a) Then at least one of v_1 and v_2 is black in $SOL^\star$.
(b) If v_1 is a cycle node in $SOL^\star$, then v_2 is also black.
(c) If v_1 and v_2 are both cycle nodes, then they belong to the same cycle in $SOL^\star$.

Proof. Part (c) follows immediately from the termination codition of Step 2. The other two parts we prove by induction on the number of iterations of Steps 2 and 3. Note that $|S| = |OPT^\star|$ at the end of Step 1. The termination condition of Step 1 implies part (b) at the end of Step 1, and part (a) if u is covered by a cycle in S at that time. If u is not covered by S and both v_1 and v_2 were not black after Step 1, we could add the sharing (v_1, u, v_2) to S and get a CMSS solution better than $OPT^\star$, which is impossible.

In Steps 2 and 3 we do not destroy a path, and we do not create new black nodes or cycle nodes. However, it may happen that we break a cycle into a path, thus turning two black nodes into gray nodes (the endpoints of the path). If a tail ends at one of these gray nodes, Step 3(a) does not terminate. This implies that part (b) holds after each iteration of Steps 2 and 3.

To prove part (a), assume there is a sharing (v_1, u, v_2) with u not covered by a path at the end of an iteration of Steps 2 and 3. Since part (a) holds at the beginning of the iteration, at least one of the two lower nodes must have been black at that time. Since only cycle nodes can change their color during an iteration, it was a cycle node. By part (b), the other node must also have been black before the iteration. Since both nodes became gray, they are the two endpoints of the path created in the iteration. But then we would add the sharing (v_1, u, v_2) to S in Step 3(b), making both nodes black again. □

Now we have all the ingredients to prove our main result. Let SOL denote the final MSS solution obtained by the "Cycle-breaking Algorithm" on G.

Theorem 7. *SOL is a $\frac{5}{3}$-approximate MSS solution.*

Proof. We partition the paths in SOL into three sets: (i) SOL_1, the paths that exist right after Step 1, (ii) SOL_2, the paths created in Steps 2 and 3, and (iii) SOL_4, the paths created in Step 4. We denote the number of sharings in SOL_i by s_i, for $i = 1, 2, 4$, and we denote the number of paths in SOL_i by p_i. Each path in SOL_1 or SOL_2 is a path in $SOL^\star$, and each path in SOL_4 corresponds to a cycle in $SOL^\star$ with the same set of lower nodes.

Let OPT denote an optimal MSS solution. We partition the sharings in OPT into three disjoint subsets. 1) The set OPT_{1+2} of all sharings whose upper nodes are contained in the paths in SOL_1 or SOL_2. 2) The set OPT_4 of all sharings whose upper nodes are not contained in any paths in SOL_1 or SOL_2, and whose two lower nodes are contained in some paths in SOL_4. 3) The set OPT_{other} of all other sharings.

For each sharing (v_1, u, v_2) in OPT_4, in $SOL^\star$ both v_1 and v_2 are cycle nodes and u is not contained in any path. Thus, by Lemma 6 (c), v_1 and v_2 are in the same cycle in $SOL^\star$, i.e., in the same path in SOL_4. Let $s(P)$ denote the number of sharings in a simple path P. For each path P in SOL_4 there are at most $s(P)$ sharings that are in OPT_4. Summing over all paths in SOL_4, we obtain $|OPT_4| \leq s_4$.

Similarly, for each sharing (v_1, u, v_2) in OPT_{other}, u is not in any path in $SOL^\star$. Thus, if v_1 (or v_2) is a cycle node in $SOL^\star$, then by Lemma 6 (b), v_2 (or v_1) is a black node in a path in $SOL^\star$. On the other hand, if neither v_1 nor v_2 is a cycle node in $SOL^\star$, then by Lemma 6 (a) at least one of v_1 and v_2 is a black node on a path in $SOL^\star$. Hence, in either case, the sharing (v_1, u, v_2) has at least one black node on a path in $SOL^\star$, i.e., on a path in SOL_1 or SOL_2. Since a lower node can appear in at most two sharings in OPT, we have $|OPT_{\text{other}}| \leq 2 \cdot \#(\text{black nodes in a path in } SOL_1 \text{ or } SOL_2) = 2(s_1 - p_1 + s_2 - p_2)$.

Since the number of sharings in OPT_{1+2} cannot exceed the number of sharings in SOL_1 and SOL_2, we have $|OPT_{1+2}| \leq s_1 + s_2$. Altogether,we have $|OPT| = |OPT_{1+2}| + |OPT_4| + |OPT_{\text{other}}| \leq s_1 + s_2 + s_3 + 2(s_1 - p_1 + s_2 - p_2) = |SOL| + 2(s_1 - p_1 + s_2 - p_2)$.

Note that we lost p_4 sharings in Step 4 of the algorithm. Moreover, by Lemma 5, $|OPT^\star| = |SOL^\star| + p_2$. Thus, $|OPT| \leq |OPT^\star| = |SOL| + p_2 + p_4$ and therefore $|OPT| + 2 \cdot |OPT| \leq |SOL| + 2(s_1 - p_1 + s_2 - p_2) + 2(|SOL| + p_2 + p_4) = 3 \cdot |SOL| + 2(s_1 + s_2 + p_4) - 2p_1$. Since $p_4 \leq s_4$ and $p_1 \geq 0$, this implies $3 \cdot |OPT| \leq 3 \cdot |SOL| + 2(s_1 + s_2 + s_4) = 5 \cdot |SOL|$. □

The $\frac{5}{3}$ approximation ratio of our algorithm is tight, as shown by the example in Fig. 5.

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 Conference (DAC)*, pp. 363–368, 2004.

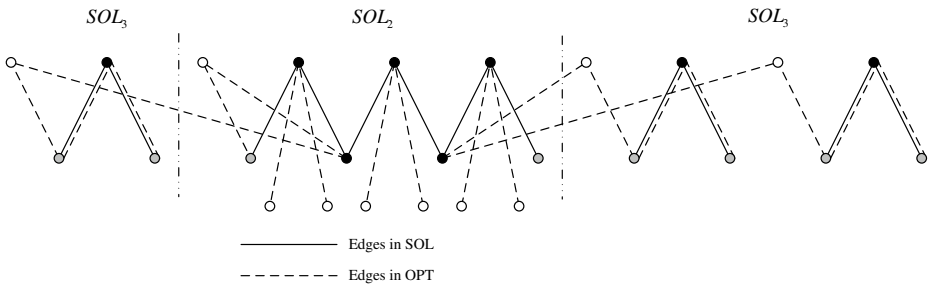

Fig. 5. An example showing that our approximation ratio $\frac{5}{3}$ is tight.

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)*, pp. 641–648, 2006.
3. A. Cao and C.-K. Koh. Non-crossing OBDDs for mapping to regular circuit structures. *Proc. IEEE International Conference on Computer Design*, pp. 338–343, 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. IEEE/ACM International Conference on Computer-Aided Design*, pp. 565–571, 2005.
5. W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS J. on Computing*, 11(2):138–148, 1999.
6. J. Edmonds. Maximum matching and a polyhedron with 0,1-nodes. *J. Res. Nat. Bur. Stand. B*, 69:125–130, 1965.
7. L. Engebretsen and M. Karpinski. TSP with bounded metrics. *Journal of Computer and System Sciences*, 72(4):509–546, 2006.
8. S. R. Kosaraju, J. K. Park, and C. Stein. Long tours and short superstrings. *Proc. 35th Annual Symp. on Foundations of Computer Science (FOCS'94)*, pp. 166–177, 1994.
9. T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Wiley, New York, 1990.
10. J. Li, A. Chaudhary, D. Z. Chen, R. Fleischer, X. S. Hu, M. T. Niemier, Z. Xie, and H. Zhu. Approximating the Maximum Sharing Problem. Submitted for publication, 2006.
11. M. Marek-Sadowska and M. Sarrafzadeh. The crossing distribution problem, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(4):423–433, 1995.
12. M. T. Niemier and P. M. Kogge. Exploring and exploiting wire-level pipelining in emerging technologies, *Proc. 28th Annual International Symp. on Computer Architecture*, pp. 166–177, 2001.
13. C. H. Papadimitriou and M. Yannakakis. The Traveling Salesman Problem with distances one and two, *Mathematics of Operations Research*, 18(1):1–11, 1993.
14. C. D. Thompson. Area-time complexity for VLSI, *Proc. 11th Annual ACM Symp. on Theory of Computing (STOC'79)*, pp. 81–88, 1979.
15. P. D. Tougaw and C. S. Lent. Logical devices implemented using quantum cellular automata, *J. of App. Phys.*, 75:1818, 1994.