Red-Blue-Partitioned MST, TSP, and Matching

Matthew P. Johnson*

Abstract

Arkin et al. [2] recently introduced *partitioned pairs* network optimization problems: Given n pairs of points in a metric space, the task is to color one point from each pair red and the other blue, and then to compute two separate *network structures* or disjoint (node-covering) subgraphs of a specified sort, one on the graph induced by the red points and the other on the blue points. Three structures have been investigated by [2]—*spanning trees*, *traveling salesperson tours*, and *perfect matchings*—and the three objectives to optimize for when computing such pairs of structures: *min-sum*, *min-max*, and *bottleneck*. We provide improved approximation guarantees and/or strengthened hardness results for these nine NP-hard problem settings.

1 Introduction

We consider the class of *partitioned pairs* network optimization problems recently introduced by Arkin et al. [2]. Given a complete metric-weighed graph G whose vertex set consists of n pairs $\{p_1, q_1\}, ..., \{p_n, q_n\}$ (with n even), the task is to color one node from each pair red and the other blue, and then to compute two *network structures* or disjoint (node-covering) subgraphs of a specified sort, one on the graph induced by the blue nodes and the other on the red nodes. One motivation is robustness: if the pairs represent n different types of resources needed to build the desired network structure, with two available instances p_i, q_i of each type i, then solving the problem means computing two separate independent instances of the desired structure, one of which can be used as a backup if the other fails.

The structures that have been investigated are *spanning trees*, *traveling salesperson*, and *perfect matchings*. A solution consists of a disjoint pair of subgraphs covering all nodes, i.e., two (partial) matchings, two trees, or two cycles, and there are different potential ways of evaluating the cost of the pair. The optimization objectives that have been considered are: 1) minimize the sum of the two structures' costs (*min-sum*), 2) minimize the maximum of the two structures' costs (*min-max*), and 3) minimize the weight of the heaviest edge used in either of the structures (*bottleneck*).

Contributions. We provide a variety of results for these nine problem settings (all of which turn out to be NP-hard; see Table 1), including algorithms with improved approximation guarantees and/or stronger hardness results for each. In particular, we provide tighter analyses of the min-sum and min-max approximation factors (along with problem instances matching these factors) of Arkin et al. [2]'s 2-MST approximation algorithm, which is equivalent to that of Algorithm 1 below. We also show that a simple extension of this algorithm (see Algorithm 2 below) provides improved min-sum and min-max approximation guarantees for 2-TSP. See the full version of the paper for omitted proofs.

Related work. The primary antecedent of this work is Arkin et al. [2] (see also references therein), which introduced the class of 2-partitioned network optimization problems. Earlier related problem settings include optimizing a path visiting at most one point from each pair [8], generalized MST [15, 17, 3], generalized TSP [3], constrained forest problems [9], adding conflict constraints to MST [18, 12, 6] and to perfect matching [16, 6], and balanced partition of MSTs [1].

2 2-MST

2.1 Min-sum/min-max 2-MST: algorithm

In this section we give a simple algorithm (see Algorithm 1) that provides an approximation guarantee for 2-MST under both the min-sum and min-max objectives. The key lemma that the approximation guarantee relies on proves a property about the result of partitioning a metric-edge-weighted spanning tree into a 2-component spanning forest.

Initially we show that *for any 2-coloring* $V_1 \cup V_2 = V$ of the graph, the sum of the costs of MSTs on V_1 and V_2 will be at most 3 times the cost of the spanning tree on V. Then we modify the argument to improve the combined cost of the two trees slightly, reducing it by the weight of their single heaviest edge in the following key lemma.

Theorem 1 *Let V^C be a set of points lying within a metric space. Let T^C be an MST on V. Let $V_1^C \cup V_2^C = V^C$ be any 2-coloring of V^C, and let T_1^C and T_2^C be MSTs of V_1^C and V_2^C, respectively. Let $e_C = \{v_L^C, v_R^C\}$ be a heaviest edge in T, with weight w_C. Let T_L, T_R be the trees (on nodes V_L^C, V_R^C, respectively) obtained by*

*Lehman College and The Graduate Center, CUNY

Table 1: Summary of results. $R, B \subseteq E$ denote the red and blue solutions, respectively. UB values indicate the approximation factors we obtain, *all for general metric spaces*; LB values indicate hardness of approximation lower bounds, *all (except min-sum and min-max TSP) for the special case of metric weights* $\{1, 2\}$. Best prior bounds (all due to [2]) are also shown, where $\rho_{\text{St}} \leq 2$ denotes the underlying metric space's Steiner ratio (conjectured to be $\frac{2}{\sqrt{3}} \approx 1.1547$ in Euc. 2D [11]), and ρ_{tsp} denotes TSP's best achievable approximation factor in the underlying metric space (currently $\rho_{\text{tsp}} = 1.5$ in general [4]).

		min-sum $c(R) + c(B)$	min-max $\max\{c(R), c(B)\}$	bottleneck $\max\{w_e : e \in B \cup R\}$
MST	**our UB**:	**3**	**4**	–
	[2]'s UB:	$(3\rho_{\text{St}})$	$(4\rho_{\text{St}})$	(9)
	our LB:	**NP-h**	**NP-h**	**2**
	[2]'s LB:	(-)	(NP-h in metric)	(-)
TSP	**our UB**:	**4**	**4**	–
	[2]'s UB:	$(3\rho_{\text{tsp}})$	$(6\rho_{\text{tsp}})$	(18)
	our LB:	$\mathbf{123/122 \approx 1.00819}$ ***with metric weights*** $\{\mathbf{.5}, \mathbf{1}, \mathbf{1.5}, \mathbf{2}\}$		**2**
	[2]'s LB:	(-)	(-)	(-)
matching	**our UB**:	–	–	–
	[2]'s UB:	(2)	(3)	(3)
	our LB:	$\mathbf{\frac{8305}{8304} \approx 1.00012}$	$\mathbf{\frac{8305}{8304} \approx 1.00012}$	**2**
	[2]'s LB:	(NP-h in metric)	(weakly NP-h in 2D Euc.)	(-)

Algorithm 1 Min-sum/min-max 2-MST approx

1: $T \leftarrow$ an MST on the $2n$ nodes
2: $\{T_L, T_R\} \leftarrow$ result of deleting a max-weight edge $e_\times$ from T
3: **for** each node pair $(p_i, q_i) \in V_L \times V_R$ **do** color p_i blue and q_i red
4: **for** each other node pair (p_i, q_i) **do** assign p_i, q_i arbitrary distinct colors
5: **for** $c \in \{b, r\}$: $T_c \leftarrow$ a minimum-weight tree spanning the color-c nodes
6: **return** $\{T_b, T_r\}$

deleting e_C from T^C, where $v_L^C \in T_L^C$ and $v_R^C \in T_R^C$. Then we have:

(a) $c(T_1^C) + c(T_2^C) \leq 3c(T^C) - w_C$, *and*

(b) $\max\{c(T_1^C), c(T_2^C)\} \leq 2c(T^C) - w_C$.

Observation 1 *There exist families of instances showing that bounds (a) and (b) of Theorem 1 are (simultaneously) tight.*

Then we analyze Algorithm 1, which forms trees T_L, T_R by deleting a max-weight edge $e_\times$ (of weight $w_\times$) from an MST T computed on the $2n$ nodes, and then colors all "lone" nodes appearing without their partners in T_L blue and all lone nodes in T_R red, and assigns arbitrary distinct colors to all other node pairs.

Theorem 2 *Algorithm 1 provides a 3-approximation for min-sum 2-MST.*

The proof analyzes three cases, depending on whether one, both, or neither T_L, T_R contains a pair, the first two cases of which imply that OPT must cross between T_L and T_R at least once or twice, respectively. The challenge is that $c(OPT)$ is lower-bounded by $c(T_L) + c(T_R)$ but not by $c(T) = c(T_L) + w_\times + c(T_R)$. We bound ALG by carefully applying Theorem 1 to *both* T_L *and* T_R, and we obtain a bound on $c(OPT)$ include $w_\times$ or $2w_\times$, permitting the two bounds to be compared, by subtracting max-weight edges from one or both sides.

This immediately implies that the same algorithm provides 6-approximation for min-max 2-MST, but we perform a tighter analysis.

Theorem 3 *Algorithm 1 provides a 4-approximation for min-max 2-MST.*

Extending Observation 1, we obtain:

Observation 2 *There exist families of instances showing that the 2-MST min-sum and min-max approximation ratios are both tight.*

2.2 Min-sum/min-max/bottleneck: hardness

We provide a reduction inspired by the reduction of [7] from Three-Dimensional Matching to the problem of partitioning a bipartite graph into two connected components, each containing exactly half the vertices.

In our reduction, however, we reduce the traditional 3-SAT problem.

Given the 3-SAT formula, we construct the following graph (see Fig. 1). For each clause, create a path of length p. For each variable x_i, we create create two nodes, x_i and $\bar{x}_i$. We also create a path of length p_b called b and a path of length p_r called r. From each x_i or $\bar{x}_i$, we draw an edge to the final nodes of the paths

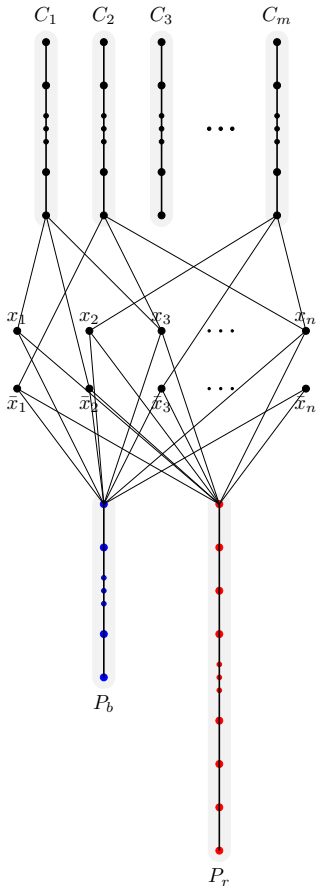


Figure 1: Spanning tree reduction.

corresponding the clauses that the literal appears in. Finally, from each x_i *and* $\bar{x}_i$, we draw edges to the final nodes paths b and r. All the edges defined have 1; all non-defined edges have weight 2. (In all cases when we refer to the "final" node of one of these $m+2$ paths, we mean the node with degree > 2.)

The path lengths are defined as follows: $p_r = (n+1) \cdot n^3 + n + n + 1, p_b = n^3 + n + 1, p = n^3 + 1$.

Then the total number of nodes in the graph constructed is: $|V| = n \cdot p + p_b + p_r + m, = 2 \cdot (n_r + m)$.

Finally, we must specify the $\{p_i, q_i\}$ pair relationships of these nodes. Each pair $\{x_i, \bar{x}_i\}$ is a $\{p_i, q_i\}$ pair. All p_r nodes of path p_r are p_i s. All p_b nodes of path p and all p nodes of path corresponding to an element are q_i nodes. Observe that results in an equal number of p_i and q_i nodes since $p_b + n \cdot p = p_r$.

Lemma 4 *The formula is satisfiable iff the constructed graph admits a 2-MST solution using only weight-1 edges.*

Thus we conclude:

Theorem 5 *In the special case of metric graphs with weights 1 and 2, min-sum and min-max, 2-MST are both (strongly) NP-Complete, and bottleneck 2-MST is NP-hard to approximate with factor better than 2.*

Algorithm 2 Min-sum/min-max 2-TSP approx

Identical to Alg, 1, except with lines 5,6 replaced by:

5: $C \leftarrow$ a TSP tour, computed from T by edge-doubling
6: **for** $c \in \{b, r\}$: $C_c \leftarrow$ a tour of the color-c nodes, computed by shortcutting C
7: **return** $\{C_b, C_r\}$

3 2-TSP

3.1 Min-sum/min-max/bottleneck 2-TSP: hardness

Clearly the min-sum and min-max objectives for 2-TSP are at least as hard to approximate as ordinary TSP in the same metric space (e.g., hard to approximate with factor better than 123/122 [13], even with edge weights $\{.5, 1, 1.5, 2\}$): to reduce TSP to either of these, simply introduce a co-located pair $\{p_v, q_v\}$ for each node v in the TSP instance. Similarly, the same reduction implies that the bottleneck objective for 2-TSP is at least as hard to approximate as ordinary bottleneck TSP in the same metric space (e.g., hard to approximate with factor better than 2, even with edge weights $\{1, 2\}$).

3.2 Min-sum/min-max 2-TSP: algorithm

Now we adapt Algorithm 1 above to obtain a 4-approximation algorithm for min-sum and min-max 2-TSP (see Algorithm 2).

The proof again analyzes three cases, depending on whether one, both, or neither T_L, T_R contains a pair. Unlike with 2-MST, 2-TSP's $c(OPT)$ *is* lower-bounded by $c(T)$ in the first two cases, and so we can compare it to the simple upper bound on $c(ALG)$ of $4c(T)$.

Theorem 6 *Algorithm 2 is a 4-approximation algorithm for min-sum 2-TSP.*

Theorem 7 *Algorithm 2 is a 4-approximation algorithm for min-max 2-TSP.*

Observation 3 *There exist families of instances showing that the 2-TSP min-sum and min-max approximation factors are both tight.*

4 2-Matching

4.1 Preliminaries

In the case of perfect matching we require that the number of pairs n be even. It will be convenient to re-express the 2-Matching problem as an equivalent problem concerning cycle covers.

We begin with some observations about the nature of feasible solutions in this setting. By definition, two nodes p_i, q_i from the same pair can never be matched

because they must receive different colors. Each must then be matched with a node of the same color, and each of *those* nodes's partners must receive the opposite color and be matched with a node of that color, and so on, in a consistent fashion. One way to make this consistency requirement concrete is the following alternative description. First, for each pair $\{p_i, q_i\}$, draw a length-2 path (of unit-weight edges) between them, separated by a dummy node d_i, and in the resulting $3n$-node graph G' consider instead the task of finding a 2-factor, i.e., a node-disjoint cycle cover, of minimum cost. In particular, consider seeking a cycle cover that uses only unit-weight edges, which would have cost $3n$.

Definition 1 *Say that a 2-matching or cycle cover is* feasible *if it uses only unit-weight edges. We call a non-dummy node of G' (i.e., a node from G) a* real node*; similarly, we call an edge between a dummy node and a real node G' a* dummy edge *and a path $p_i d_i q_i$ a* dummy path*; we call an edge between two real nodes a* real edge.

Observe that any feasible 2-matching in G will induce a 2-factor of G': imagine G' drawn in a "tripartite" style, with the red nodes in the left column, the blue nodes in the right column, and the dummy nodes in the center column. Then each path $p_i - d_i - q_i$ forms a "cross-edge" (going either left or right), each red edge appears in the left column, and each blue edge appears in the right column. Each non-dummy node is matched with one other node in the 2-matching, so if we combine the edges of the paths $p_i - d_i - q_i$ to those of the matching, then in the graph induced by these edges, each of the $3n$ nodes will have degree 2. This implies the edge set is a 2-factor. Note that the cost of the 2-factor differs by a known amount ($2n$, because each dummy nodes two edges are unit-weight)) from the (min-sum) cost of the corresponding 2-matching.

The problem of finding a minimum-cost 2-factor is known to be polynomial-time solvable by reduction to bipartite matching (folklore). Unfortunately, a 2-factor of G' will not necessarily induce a valid 2-matching on G. In G' as defined, the additional property needed (somewhat analogously to bipartite graphs having no odd cycles) is for *each cycle's size to be a multiple of 6*, which we will call a $C_{6\times}$-cover.

Definition 2 *Let a $C_{6\times}$-cover for a given graph be a 2-factor, i.e., a node-disjoint collection of subgraphs covering all nodes, where each subgrraph is a member of $\{C_6, C_{12}, C_{18}, ...\}$.*

Lemma 8 *Any feasible $C_{6\times}$-cover for G' will induce a feasible 2-matching for G.*

Unfortunately, unlike the problem of deciding whether a graph admits a feasible cycle cover, deciding whether it admits a $C_{6\times}$-cover is NP-Complete [10]. This fact does not immediately imply the hardness of the 2-Matching problems, however, because G' is not an arbitrary graph. We can characterize it as follows. It contains $3n$ nodes consisting of n triples $\{p_i, d_i, q_i\}$, where each d_i is degree 2, with neighbors p_i, q_i.

4.2 Bottleneck 2-Matching: hardness

To prove hardness, we give a reduction inspired by Papadimitriou's reduction [5] from 3-SAT to the problem of deciding whether a graph can be partitioned into a node-disjoint collection of cycles, each of size *at least 6*.

We reduce from Monotone 1-in-3 SAT (which has no negated literals) to the problem of deciding whether G' admits a (feasible, i.e., using unit-weight edges only) $C_{6\times}$-cover. Recall that edge weights in G' are 1 or 2, and that each dummy node's two edges are weight-1. Given the boolean formula, we proceed as follows.

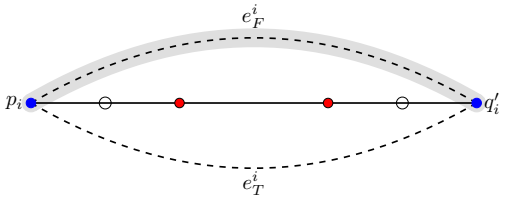
For each variable x_i, we create a gadget as shown in Fig. 2a. It consists of a 6-path $(p_i, d_i, q_i, p'_i, d'_i, q'_i)$, whose nodes form two triples $\{p_i, d_i, q_i\}, \{p'_i, d'_i, q'_i\}$, plus an edge (p_i, q'_i) labeled e^i_T and a *pseudoedge* labeled e^i_F. There will be exactly two feasible ways to cover the nodes of this gadget in a $C_{6\times}$-cover, with the cycle including e^T_i, corresponding to x_i being true, and the one including e^F_i, corresponding to false.

For each clause C_j, we create a gadget as shown in Fig. 2b. It consists of two copies of K_4, where each node u^j_ℓ in one K_r is connected by a 2-path and dummy node to a corresponding node v^j_ℓ in the other. Three *pseudoedges* connecting a distinguished node u^j_0 to the other three nodes of the same K_4 are labeled f^j_1, f^j_2, f^j_3. If a feasible $C_{6\times}$-cover, one of these edges will be on and the other two off, corresponding to a satisfied 1-in-3 SAT clause.

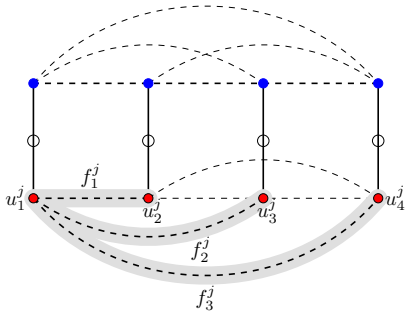
Definition 3 *A* pseudoedge *is an edge, or the result of attaching a connection gadget to a pseudoedge.*

Finally, to implement the appearance of a variable in a clause, we use the gadget shown in Fig. 2c, which will appear in sequence. Applying a connection gadget to pseudoedges e^i_F and f^j_ℓ does the following:

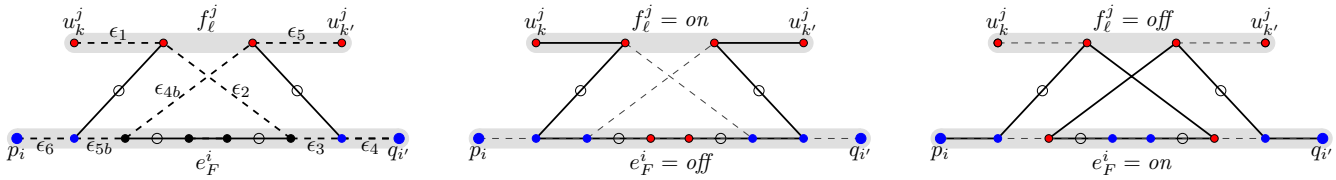
1. the last (rightmost) edge of f^j_ℓ is split into a 9-path path via the creation of 8 new nodes (compare e^i_F in Figs. 2a, 2c(left), and 2d);
2. f^j_ℓ's edge is replaced with two new edges (labeled ϵ_1, ϵ_5 in Fig. 2c) incident to two new nodes (compare f^j_ℓ in Figs. 2b and 2c(left));
3. f^j_ℓ's first new node is connected to e^i_F's first and seventh new nodes, by a 2-path and an edge, respectively (see Fig. 2c(left)); and
4. f^j_ℓ's second new node is connected to e^i_F's second and eighth new nodes, by an edge and a 2-path, respectively (see Fig. 2c(left)).



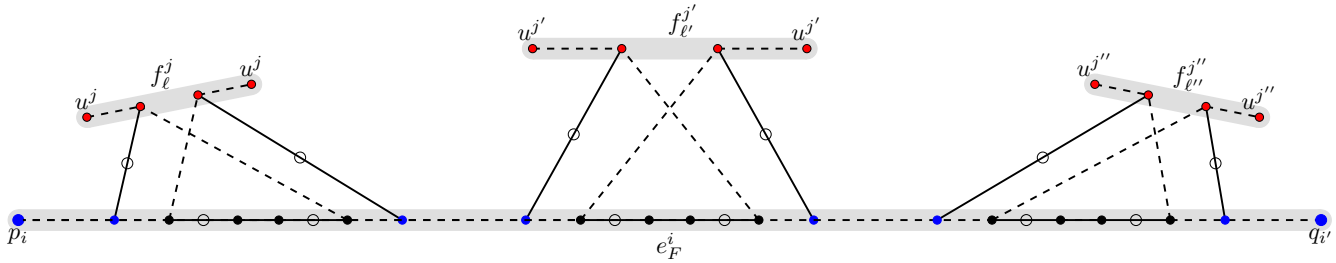
(a) Variable gadget for x_i. Any feasible $C_{6\times}$-cover must include pseudoedge e^i_F xor edge e^i_T.



(b) Clause gadget for C_j. Any feasible $C_{6\times}$-cover must include *exactly one* of the three distinguished pseudoedges f_1, f_2, f_3 (plus one of the unlabeled dashed edges from the bottom and two of the top).



(c) Connection gadget (left fig.) for an appearance (negated iff $v = F$) of variable x_i in clause C_j. The lower shaded path is a more detailed view of one of the x_i gadget's pseudoedge e^i_F (see (a)); the upper shaded path is a more detailed view of one of the C_j gadget's three distinguished pseudoedges f^j_1, f^j_2, f^j_3 (see (b)). We show (see Lemma 8) that there are only two possible feasible $C_{6\times}$-covers of the gadget, one in which f^j_ℓ is *on* and e^i_F is *off*, meaning this connection represents C_j's unique true literal (middle fig.), and one in which f^j_ℓ is *off* and e^i_F is *on*, meaning it represents one of C_j's two false literals (right fig.).



(d) The example e^i_F shown here is a more detailed view of pseudoedge e^i_F in variable x_i's gadget (see (a)). An e^i_F can have multiple connections (*in this example*, three), corresponding to appearances of x_i's in different clauses (in this example, an x_i literal appears as the ℓth literal in clause C_j, and so on; typically j, j', j'' will *all be distinct*); an f^j_ℓ has only one connection, since it indicates what literal the ℓth literal in clause C_j is. Subscripts of u nodes are omitted for clarity.

Figure 2: Gadgets used in 2-Matching's hardness proof. Real nodes are shown filled in, dummy nodes unshaded. Edges that must be used in any feasible solution are shown solid, other edges dashed. e^i_F and f^j_1, f^j_2, f^j_3 are pseudoedges, i.e., schematic representations of paths that connections attach to.

For each variable x_i appearing (in some position $k \in [3]$) within a clause C_j, we draw a connection gadget between x_i's e^i_F and C_j's f^j_k. First observe the following, which can be verified by inspection:

Fact 1 *If all pseudoedges e^i_F and f^j_k were simply edges, then a $C_{6\times}$-cover would induce one of two legal states within any variable X_i's gadget, with exactly one of e^i_F, e^i_T on, and one of three legal states within any clause C_j's gadget, with exactly one of f^j_1, f^j_2, f^j_3 on.*

Now we show that any $C_{6\times}$-cover will induce one of two canonical states on each connection gadget (see Fig. 2c middle and right), each pseudoedge, and each variable gadget.

Lemma 9 *Within any pseudoedge pair (e^i_F, f^j_k) connected by a connection gadget, a feasible $C_{6\times}$-cover induces one of only two legal states: one with the first and last edges (labeled ϵ_1 and ϵ_5, respectively, in Fig. 2c(left)) within f^j_k on ("f^j_k is on"), and the other with with the first and last edges (labeled ϵ_6 and ϵ_4, respectively, in Fig. 2c(left)) within e^i_F on ("e^i_F is on").*

This immediately implies:

Corollary 1 *A feasible $C_{6\times}$-cover induces one of two canonical states within each variable gadget and one of three canonical states within each cause gadget.*

In a solution where the clause's edge f_k^j is on, this forces $e_{i_k}^F$ to be off, and hence $e_{i_k}^T$ to be on; similarly, it forces clause C_j's other two distinguished pseudoedges to be off, and hence the variables connected to those edges to be false. (The clause gadget's other edges can be freely used or not, as needed to form a feasible $C_{6\times}$-cover.) Finally, observe that the final constructed graph G' indeed satisfies the required structure for corresponding to an equivalent instance G of the 2-Matching problem: every dummy node has exactly two neighbors (both real), and every real node has exactly one dummy neighbor.

From the arguments above, we conclude that G' admits an all-unit weight $C_{6\times}$-cover iff G admits an all-unit weight 2-matching iff the underlying boolean formula is satisfiable. Thus we conclude:

Theorem 10 *In the special case of metric graphs with weights 1 and 2, bottleneck 2-Matching is NP-hard to approximate with factor better than 2 (and min-sum and min-max 2-Matching are both (strongly) NP-Complete).*

4.3 Min-sum/min-max 2-Matching: hardness

By reduction from a special case of Max 1-in-3 SAT, we can obtain a hardness of approximation result for the min-sum and min-max objectives. Let Max 1-in-3 SAT-5 denote Max 1-in-3 SAT under the restriction that each variable appears in at most 5 clauses.

Lampis has shown (implicitly in [14][1]) the following:

Lemma 11 *There exists a family of* Max 1-in-3 SAT-5 *instances with $15m$ clauses and $8.4m$ variables, each appearing in at most 5 clauses, for which, for any $\epsilon > 0$, it is NP-hard to decide whether the minimum number of unsatisfiable clauses is at most ϵm or at least $(0.5 - \epsilon)m$.*

For concreteness, let Min Not-1-in-3 SAT-5 indicate the optimization problem of minimizing the number of unsatisfied clauses in a 1-in-3 SAT-5 formula.

Now we argue that the same construction used above provides an approximation-preserving reduction from Min Not-1-in-3 SAT-5.

Corollary 2 *Min-sum and min-max 2-Matching are both, in the special case of metric graphs with weights 1 and 2, NP-hard to approximate with factor better than $8305/8304 \approx 1.00012$.*

[1] Karpinksi et al. [13] provide a similar construction yielding a stronger hardness of approximation lower bound for Metric TSP, but adapting that construction to our present problem actually leads to a slightly weaker lower bound.

Acknowledgements. This work was supported in part by NSF award INSPIRE-1547205, and by the Sloan Foundation via a CUNY Junior Faculty Research Award. We thank Ali Assapour, Ou Liu, and Elahe Vahdani for useful discussions.

References

[1] M. Andersson, J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Balanced partition of minimum spanning trees. *International Journal of Computational Geometry & Applications*, 13(04):303–316, 2003.

[2] E. M. Arkin, A. Banik, P. Carmi, G. Citovsky, S. Jia, M. J. Katz, T. Mayer, and J. S. B. Mitchell. Network optimization on partitioned pairs of points. In *ISAAC*, pages 6:1–6:12, 2017.

[3] B. Bhattacharya, A. Ćustić, A. Rafiey, A. Rafiey, and V. Sokol. Approximation algorithms for generalized MST and TSP in grid clusters. In *COCOA*, pages 110–125. 2015.

[4] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 88, Management Sciences Research Group, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[5] G. Cornuejols and W. Pulleyblank. A matching problem with side conditions. *Discrete Mathematics*, 29(2):135–159, 1980.

[6] A. Darmann, U. Pferschy, J. Schauer, and G. J. Woeginger. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726–1735, 2011.

[7] M. E. Dyer and A. M. Frieze. On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics*, 10(2):139–153, 1985.

[8] H. N. Gabow, S. N. Maheshwari, and L. J. Osterweil. On two problems in the generation of program test paths. *IEEE Transactions on Software Engineering*, (3):227–231, 1976.

[9] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[10] P. Hell and D. G. Kirkpatrick. Packings by cliques and by finite families of graphs. *Discrete Mathematics*, 49(1):45–59, 1984.

[11] A. O. Ivanov and A. A. Tuzhilin. The Steiner ratio Gilbert–Pollak conjecture is still open. *Algorithmica*, 62(1-2):630–632, 2012.

[12] M. M. Kanté, C. Laforest, and B. Momege. Trees in graphs with conflict edges or forbidden transitions. In *TAMC*, pages 343–354. Springer, 2013.

[13] M. Karpinski, M. Lampis, and R. Schmied. New inapproximability bounds for TSP. *Journal of Computer and System Sciences*, 81(8):1665–1677, 2015.

[14] M. Lampis. Improved inapproximability for TSP. In *APPROX/RANDOM*, pages 243–253. Springer, 2012.

[15] Y.-S. Myung, C.-H. Lee, and D.-W. Tcha. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241, 1995.

[16] T. Öncan, R. Zhang, and A. P. Punnen. The minimum cost perfect matching problem with conflict pair constraints. *Computers & Operations Research*, 40(4):920–930, 2013.

[17] P. C. Pop. New models of the generalized minimum spanning tree problem. *Journal of Mathematical Modelling and Algorithms*, 3(2):153–166, 2004.

[18] R. Zhang, S. N. Kabadi, and A. P. Punnen. The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization*, 8(2):191–205, 2011.