Approximability of Connected Factors

Kamiel Cornelissen[1], Ruben Hoeksma[1], Bodo Manthey[1],
N. S. Narayanaswamy[2], and C. S. Rahul[2]

[1]University of Twente, Enschede, The Netherlands
{k.cornelissen, r.p.hoeksma, b.manthey}@utwente.nl
[2]Indian Institute of Technology Madras, Chennai, India
{swamy, rahulcs}@cse.iitm.ac.in

Finding a d-regular spanning subgraph (or d-factor) of a graph is easy by Tutte's reduction to the matching problem. By the same reduction, it is easy to find a minimal or maximal d-factor of a graph. However, if we require that the d-factor is connected, these problems become NP-hard – finding a minimal connected 2-factor is just the traveling salesman problem (TSP).

Given a complete graph with edge weights that satisfy the triangle inequality, we consider the problem of finding a minimal connected d-factor. We give a 3-approximation for all d and improve this to an $(r+1)$-approximation for even d, where r is the approximation ratio of the TSP. This yields a 2.5-approximation for even d. The same algorithm yields an $(r+1)$-approximation for the directed version of the problem, where r is the approximation ratio of the asymmetric TSP. We also show that none of these minimization problems can be approximated better than the corresponding TSP.

Finally, for the decision problem of deciding whether a given graph contains a connected d-factor, we extend known hardness results.

1 Introduction

The traveling salesman problem (Min-TSP) is one of the basic combinatorial optimization problems: given a complete graph $G = (V, E)$ with edge weights that satisfy the triangle inequality, the goal is to find a Hamiltonian cycle of minimum total weight. Phrased differently, we are looking for a subgraph of G of minimum weight that is 2-regular, connected, and spanning. While Min-TSP is NP-hard [12, ND22], omitting the requirement that the subgraph must be connected makes the problem polynomial-time solvable [19, 27]. In general, d-regular, spanning subgraphs (also called d-factors) of minimum weight can be found in polynomial time using Tutte's reduction [19, 27] to the matching problem. Cheah and Corneil [6] have shown that deciding whether a given graph $G = (V, E)$ has a d-regular connected spanning subgraph is NP-complete for every $d \geq 2$, where $d = 2$ is just the Hamiltonian cycle problem [12, GT37]. Thus, finding a connected d-factor of minimum weight is also NP-hard for all d.

While one might think at first glance that the problem cannot become easier for larger d, finding (minimum-weight) connected d-factors is easy for $d \geq n/2$, where $n = |V|$, as in this case any d-factor is already connected. This poses the question for which values of d (as a function of n) the problem becomes tractable.

In this paper, we analyze the complexity and approximability of the problem of finding a d-factor of minimum weight.

1.1 Problem Definitions and Preliminaries

In the following, n is always the number of vertices. To which graph n refers will be clear from the context.

All problems defined below deal with undirected graphs, unless stated otherwise. For any d, d-RCS is the following decision problem: Given an arbitrary undirected graph G, does G have a connected d-factor? Here, d can be a constant, but also a function of the number n of vertices of the input graph G. 2-RCS is just the Hamiltonian cycle problem.

Just as Min-TSP is the optimization variant of 2-RCS, we consider the optimization variant of d-RCS, which we call Min-d-RCS: As an instance, we are given an undirected complete graph $G = (V, E)$ and non-negative edge weights w that satisfy the triangle inequality, i.e., $w(\{x, z\}) \leq w(\{x, y\}) + w(\{y, z\})$ for every $x, y, z \in V$. The goal of Min-d-RCS is to find a connected d-factor of G of minimum weight. Min-2-RCS is just Min-TSP.

A *bridge edge* of a graph is an edge whose removal increases the number of components of the graph. A graph G is called *2-edge connected* if G is connected and does not contain bridge edges. For even d, any connected d-factor is also 2-edge-connected, i.e., does not contain bridge edges. This is not true for odd d. If we require 2-edge-connectedness also for odd d, we obtain the problem Min-d-R2CS, which is defined as Min-d-RCS, but asks for a 2-edge-connected d-factor. For consistency, Min-d-R2CS is also defined for even d, although it is then exactly the same problem as Min-d-RCS.

Finally, we also consider the asymmetric variant of the problem: given a directed complete graph $G = (V, E)$, find a spanning connected subgraph of G that is d-regular. Here, d-regular means that every vertex has indegree d and outdegree d. We denote the corresponding minimization problem by Min-d-ARCS. Min-1-ARCS is just the asymmetric TSP (Min-ATSP).

Max-d-RCS and Max-d-ARCS are the maximization variants of Min-d-RCS and Min-d-ARCS, respectively. For Max-d-RCS and Max-d-ARCS we do not require that the edge weights satisfy the triangle inequality. In the same way as for the minimization variants, Max-2-RCS is the maximum TSP (Max-TSP) and Max-1-ARCS is the maximum ATSP (Max-ATSP).

If the graph and its edge weights are clear from the context, we abuse notation by also denoting by d-RCS a minimum-weight connected d-factor, by d-R2CS a minimum-weight 2-edge-connected d-factor, and by d-ARCS a minimum-weight connected d-regular subgraph of a directed graph.

In the same way, let d-F denote a minimum-weight d-factor (no connectedness required) of a graph and let d-AF denote a minimum-weight d-factor of a directed graph. Let MST denote a minimum-weight spanning tree, and let TSP and ATSP denote minimum-weight (asymmetric) TSP tours. We have 2-RCS = TSP and 1-ARCS = ATSP. Furthermore, 2-F is the undirected cycle cover problem and 1-AF is the directed cycle cover problem.

We note that d-factors do not exist for all combinations of d and n. If both n and d are odd, then no n-vertex graph possesses a d-factor. For all other combinations of n and d with $d \leq n - 1$, there exist d-factors in n-vertex graphs, at least in the complete graph.

In the following, K_n denotes the undirected complete graph on n vertices. A vertex v of a graph G is called a *cut vertex* if removing v increases the number of components of G.

1.2 Previous Results

Requiring connectedness in addition to some other combinatorial property has already been studied for dominating sets [14] and vertex cover [9]. For problems such as minimum s-t vertex separator, which are known to be solvable in polynomial time, the connectedness condition makes it NP-hard, and recent results have studied the parameterized complexity of finding a connected s-t vertex separator [20]. Also finding connected graphs with given degree sequences that are allowed to be violated only slightly has been well-studied [5, 26].

As far as we are aware, so far only the maximization variant Max-d-RCS of the connected factor problem has been considered for $d \geq 3$. Baburin, Gimadi, and Serdyukov proved that Max-d-RCS can be approximated within a factor of $1 - \frac{2}{d \cdot (d+1)}$ [2, 13]. A slightly better approximation ratio can be achieved if the edge weights are required to satisfy the triangle inequality [3]. Baburin and Gimadi also considered approximating both Max-d-RCS and Min-d-RCS (both without triangle inequality) for random instances [3, 4]. For $d = 2$, we inherit the approximation results for Min-TSP of 3/2 [29, Section 2.4] and Max-TSP of 7/9 [21]. For $d = 1$, we inherit the $O(\log n / \log \log n)$-approximation for Min-ATSP [1] and 2/3 for Max-ATSP [15]. As far as we know, no further polynomial-time approximation algorithms with worst-case guarantees are known for Min-d-RCS. Like for Min-TSP [29, Section 2.4], the triangle inequality is crucial for approximating Min-d-RCS and Min-d-ARCS – otherwise, no polynomial-time approximation algorithm is possible, unless P = NP. Baburin and Gimadi [2, 3] claimed that Max-d-RCS is APX-hard because it generalizes Max-TSP. However, this is only true if we consider d as part of the input, as then $d = 2$ corresponds to Max-TSP.

1.3 Our Results

Table 1 shows an overview of previous results and our results.

Our main contributions are a 3-approximation algorithm for Min-d-RCS for any d and a 2.5-approximation algorithm for Min-d-RCS for even d (Section 3). The latter is in fact an $(r + 1)$-approximation algorithm for Min-d-RCS, where r is the factor within which Min-TSP can be approximated. This result can be extended to Min-d-ARCS, where r is now the approximation ratio of Min-ATSP. Our approximation algorithms, in particular for the maximization variants, are in the spirit of the classical approximation algorithm of Fisher et al. [11] for Max-TSP: compute a non-connected structure, and then remove and add edges to make it connected.

As lower bounds, we prove that Min-d-RCS and Min-d-ARCS cannot be approximated better than Min-TSP and Min-ATSP, respectively (Section 4). In particular, this implies the APX-hardness of the problems.

We prove some structural properties of connected d-factors and their relation to TSP, MST, and d-factors without connectedness requirement (Section 2). Some of these properties are needed for the approximation algorithms and some might be interesting in their own right or were initially counterintuitive to us.

Our algorithms work for all values of d, even when d is part of the input. The hardness results are extended to the case where d grows with n. In Section 5, we improve our approximation guarantee for $d \geq n/3$, prove that $(\frac{n}{2} - 1)$-RCS $\in$ P, and generalize Baburin and Gimadi's algorithm [2] to directed instances.

3

problem	result	reference
d-RCS	in P for $d \geq \frac{n}{2} - 1$ NP-complete for constant d and d of any growth rate up to $O(n^{1-\varepsilon})$	trivial for $d \geq n/2$, Section 5.2 Cheah and Corneil [6] Section 4.2
Min-d-RCS	$(r+1)$-approximation for even d 3-approximation for odd d 2-approximation for $d \geq n/3$ no better approximable than Min-TSP	Section 3.2 Section 3.1 Section 5.1 Section 4.1
Min-d-R2CS	3-approximation no better approximable than Min-TSP	Section 3.1 Section 4.1
Min-d-ARCS	$(r+1)$-approximation no better approximable than Min-ATSP	Section 3.2 Section 4.1
Max-d-RCS	$(1 - \frac{2}{d \cdot (d+1)})$-approximation	Baburin and Gimadi [2]
Max-d-ARCS	$(1 - \frac{1}{d \cdot (d+1)})$-approximation	Section 5.3

Table 1: Overview of the complexity and approximability of finding (optimal) connected d-factors. We left out that all optimization variants are polynomial-time solvable for $d \geq n/2$ and APX-hard according to Sections 4.1 and 4.2. Here, r is the approximation ratio of Min-TSP or Min-ATSP.

2 Structural Properties

In the following two lemmas, we make statements about the relationship between the weights of optimal solutions of the different minimization problems. We call an inequality $A \leq c \cdot B$ *tight* if, for every $\varepsilon > 0$, replacing c by $c - \varepsilon$ does not yield a valid statement for all instances.

Lemma 2.1 (undirected comparison).   *1. $w(\mathsf{MST}) \leq w(d\text{-RCS}) \leq w(d\text{-R2CS})$ for all d and all undirected instances, and this is tight.*

2. *$w(d\text{-F}) \leq w(d\text{-RCS})$ for all d and all undirected instances, and this is tight.*

3. *$w(d\text{-R2CS}) \leq 3 \cdot w(d\text{-RCS})$ for all odd d and all undirected instances, and this is tight for all odd d.*

4. *$w(\mathsf{TSP}) \leq w(d\text{-RCS})$ for all even d and all undirected instances, and this is tight.*

5. *$w(\mathsf{TSP}) \leq 2 \cdot w(d\text{-RCS})$ for all odd d and all undirected instances, and this is tight for all odd d.*

6. *$w(\mathsf{TSP}) \leq \frac{4}{3} \cdot w(3\text{-R2CS})$ for all undirected instances, and this is tight.*

7. *For all odd d, there are instances with $w(\mathsf{TSP}) \geq (\frac{4}{3} - o(1)) \cdot w(d\text{-R2CS})$.*

8. *$w((d-2)\text{-F}) \leq \frac{d-2}{d} \cdot w(d\text{-F})$ and $w((d-2)\text{-RCS}) \leq w(d\text{-RCS})$ for all even $d \geq 4$ and all undirected instances, and both inequalities are tight.*

4

9. *Monotonicity does not hold for odd d: for every odd $d \geq 5$, there exist instances with $w((d-2)\text{-RCS}) \geq \frac{d+2}{d} \cdot w(d\text{-RCS})$.*

Proof. Items 1 and 2: The inequalities follow immediately from the definitions. The inequality of Item 2 and the second inequality of Item 1 are tight for instances where all edge weights are equal. That the first inequality of Item 1 is also tight can be seen as follows: For any $k \geq 2$ we can construct a graph consisting of k groups of $d+1$ vertices. We set the distance between each pair of vertices from the same group equal to 0, and the distance between each pair of vertices from different groups equal to 1. Any MST of the new instance has a weight of $k-1$. We can construct a (2-edge-)connected d-factor of weight k by combining a global TSP tour with a $d-2$-factor within each group. Since this holds for all k, the first inequality of Item 1 is tight.

Item 3: The inequality is shown to hold constructively by Algorithm 1, which computes a 2-edge-connected d-factor that weighs no more than three times the weight of a minimum-weight connected d-factor. It is tight because of the following example: The set of vertices of the instance consists of a vertex v plus d sets $V_1, \ldots, V_d$ which consist of $d+2$ vertices each. We set the distance of v to each of the other vertices equal to 1. The distance between each pair of vertices from the same set V_i is 0. Finally, the distance between all pairs of vertices from different sets V_i and V_j is 2. An optimal connected d-factor connects v to one vertex of each V_i. Since each V_i has an odd number of vertices and d is odd as well, we can complete the connected d-factor without any further cost. Thus, the total cost is d.

An optimal 2-edge-connected d-factor has a weight of $3d$: Because each V_i has an odd number of vertices and d is odd, any 2-edge-connected d-factor must have at least three edges leaving each set V_i. If such an edge e is incident with v, then we charge its weight to V_i. The other possibility is that e is incident with a vertex from some V_j, where $j \neq i$. In this case e has a weight of 2 and we charge a weight of 1 to both V_i and V_j. The total charge of all sets V_I equals the total weight of the 2-edge-connected d-factor. Since each V_i is charged at least 3, the total weight of any 2-edge-connected d-factor is at least $3d$.

Item 4: Since d is even, d-RCS is Eulerian and we can take shortcuts to obtain a TSP tour whose weight is no larger than w(d-RCS). This is tight because $w(\mathsf{TSP}) \geq w(\mathsf{MST})$ and Item 1.

Item 5: If we duplicate each edge of a connected d-factor, we obtain a Eulerian multi-graph. Taking shortcuts yields a TSP tour and proves the inequality. The same instance as we used in the proof of Item 3 shows that this is tight, as every set V_i is charged at least 2 for any TSP tour.

Item 6: We exploit the following property of 3-regular 2-edge-connected graphs: for every odd subset U of the vertices, there are at least three edges that connect U to $V \setminus U$. This implies that the fractional edge-coloring number of such a graph is 3 [24, Corollary 28.5a]. We consider an optimal solution 3-R2CS. The above implies that there exists a collection of – not necessarily disjoint – matchings $M_1, \ldots, M_k$ for some k as well as non-negative numbers $\lambda_1, \ldots, \lambda_k$ such that $\sum_{i:e \in M_i} \lambda_i = 1$ for all edges $e \in$ 3-R2CS and $\sum_{i=1}^k \lambda_i = 3$ (Seymour [25] attributes this to Edmonds [8]). This implies $\sum_{i=1}^k \lambda_i w(M_i) = w(\text{3-R2CS})$. Since the total number of edges in 3-R2CS equals $3n/2$ and the λ_i's sum to 3, all the matchings $M_1, \ldots, M_k$ are necessarily perfect. Also, since $\sum_{i=1}^k \lambda_i = 3$, there exists an i with $w(M_i) \leq \frac{1}{3} \cdot w(\text{3-R2CS})$. Let M be a minimum-weight perfect matching of the instance. Then $w(M) \leq w(M_i)$. Furthermore, adding M to 3-R2CS yields a 4-regular connected multi-graph of weight at most $\frac{4}{3} \cdot w(\text{3-R2CS})$, which is thus Eulerian. Taking shortcuts yields a TSP tour, which proves the claim. It is tight by Item 7.
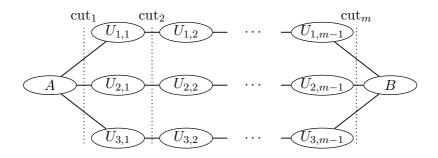
Figure 1: Graph with $w(\mathsf{TSP}) \geq (\frac{4}{3} - o(1)) \cdot w(d\text{-}\mathsf{R2CS})$. Gadgets A and B consist of $d+2$ vertices, all $U_{i,j}$ consist of $d+1$ vertices. Edges between a pair of vertices in the same gadget have weight 0. An edge between two gadgets indicates that the weight of an edge between a vertex in the first gadget and a vertex in the second gadget equals 1. All other distances are obtained via metric completion.

Item 7: For arbitrary odd d we construct a complete graph $G = (V, E)$ as follows: G consists of $3m - 1$ gadgets A, B, and $U_{i,j}$ for $i \in \{1, 2, 3\}$ and $j \in \{1, \ldots, m-1\}$. The gadgets A and B consist of $d + 2$ vertices, all $U_{i,j}$ consist of $d + 1$ vertices. All pairs of vertices within the same gadget have a distance of 0. All edges between A and $U_{i,1}$, between $U_{i,j}$ and $U_{i,j+1}$, and between $U_{i,m-1}$ and B have a weight of one for all $i \in \{1, 2, 3\}$ and $j \in \{1, \ldots, m-2\}$. All other distances are obtained by metric completion, i.e., by taking the shortest path distances. Thus, e.g., the distance between a vertex in A and a vertex in B is m. Figure 1 depicts the construction.

We build a d-factor of total weight $3m$ as follows: For each pair of gadgets that has a distance of 1, we take an edge between the two gadgets and include it in the d-factor. We do so in a way that all selected edges of weight 1 are disjoint. We complete the d-factor by taking appropriate edges of weight 0 within the gadgets. By the choice of the size of the gadgets, this can be done. The d-factor obtained is even 2-edge-connected.

Now we show that $w(\mathsf{TSP}) \geq 4m - 2$. Let T be a minimum-weight TSP tour on G. Consider T', the multigraph where each edge in T of length greater than 0 is replaced by its shortest path over edges with weight 1. Then $w(T') = w(T)$. For $j \in \{2, \ldots, m-1\}$, let cut_j be the sum over all i of the number of edges connecting $U_{i,j-1}$ to $U_{i,j}$. Let cut_1 be the number of edges connecting A to $U_{1,1}$, $U_{2,1}$, and $U_{3,1}$, and let cut_m be the number of edges connecting $U_{1,m-1}$, $U_{2,m-1}$, and $U_{3,m-1}$ to B. (The cuts are indicated by dotted lines in Figure 1.) Since T' uses only edges within gadgets or edges of weight 1, we have $w(T') = \sum_{j=1}^{m} \mathrm{cut}_j$. Since T' is Eulerian, we know that cut_j is even for all j. Since T' is connected, $\mathrm{cut}_j \geq 2$ for all j. If $\mathrm{cut}_j \geq 4$ for all j, then $w(T') \geq 4m - 2$, and we are done. Otherwise, $\mathrm{cut}_j = 2$ for some j. For ease of notation, we assume that $j \in \{2, \ldots, m-1\}$. The remaining cases are almost identical. Then there is some $U_{i,j-1}$ that is not connected to $U_{i,j}$ in T'. Thus, there must be two paths in T' that connect $U_{i,j-1}$ via A to $U_{i',j-1}$ and $U_{i'',j-1}$ ($i', i'' \neq i$, but $i' = i''$ is allowed). In the same way, T' must contain two paths from $U_{i,j}$ via B to $U_{i',j}$ and $U_{i'',j}$. The weight of the former paths is $2j - 2$ each. The weight of the latter paths is $2m - 2j$ each. In total, the weight is $4m - 4$. Adding cut_j yields the result.

Item 8: The inequality $w((d-2)\text{-}\mathsf{F}) \leq \frac{d-2}{d} \cdot w(d\text{-}\mathsf{F})$ holds since every d-factor for even d can be split into $d/2$ 2-factors, and we can remove the lightest 2-factor to obtain a $(d-2)$-factor. This is tight if all edge weights are equal.

Now consider the claim for connected factors. For $n = d + 1$, the d-factor is the complete graph and the claim is trivial. Thus, we assume $n > d + 1$. Let R be a connected d-factor. We describe a process to obtain a connected $(d-2)$-factor from R that weighs at most the weight of R. To do this, we use the following invariants: First, if a graph has only even degrees and is connected, then it is 2-edge-connected. (Otherwise, removing a bridge edge would result in two components, and the sum of degrees within each component would be odd.) Second, if the maximum degree of a graph is $d \leq n - 2$ and the graph is connected, then any vertex with degree d is adjacent to at least two vertices x and y such that x and y are not adjacent.

Let us now describe the process: We take any vertex v with degree d. If v is not a cut vertex, then we choose a pair of edges $\{v, x\}$ and $\{v, y\}$ of R such that $\{x, y\} \notin R$. This exists due to the invariant. We replace $\{v, x\}$ and $\{v, y\}$ by $\{x, y\}$, which reduces the degree of v by two and does not change the degree of the other vertices. The graph is still connected, as v has still a degree of at least $d - 2 \geq 2$ and was no cut vertex.

If v is a cut vertex, let $C_1, \ldots, C_k$ be the components obtained by removing v. Vertex v has at least two edges to each component by 2-edge-connectedness. We take one edge to $x \in C_1$ and one edge to $y \in C_2$ and shortcut it, i.e, remove $\{v, x\}$ and $\{v, y\}$ and add $\{x, y\}$. Again this reduces the degree of v by two and does not change the degree of any other vertex. Also by 2-edge-connectedness, it does not disconnect the graph. Repeating this process until all vertices have degree $d - 2$ yields a connected $(d-2)$-factor. By the triangle inequality, its weight is no more than the weight of the connected d-factor.

The analysis is tight because of Item 4.

Item 9: Consider the following instance: we have a central vertex v and d sets $V_1, \ldots, V_d$, which each consist of $d + 2$ vertices. The distance of v to each other vertex is 1. Within any set V_i, the distances are 0. The distance between each pair of vertices in two different sets V_i and V_j is 2.

The optimal connected d-factor has a weight of d: we connect v to one vertex of each set V_i and complete the d-factor locally within each V_i. In any connected $(d-2)$-factor, v is connected to $k \leq d - 2$ of the sets $V_1, \ldots, V_d$. Thus, there are $d - k$ sets that have to be connected with an edge of weight 2. This results in costs of at least $k + 2 \cdot (d - k) \geq d + 2$. $\qquad \square$

Lemma 2.2 (directed comparison). *1. $w(d\text{-AF}) \leq w(d\text{-ARCS})$ for all d and all directed instances, and this is tight.*

    2. $w(\text{ATSP}) \leq w(d\text{-ARCS})$ for all d and all directed instances, and this is tight.

    3. $w((d-1)\text{-AF}) \leq \frac{d-1}{d} \cdot w(d\text{-AF})$ and $w((d-1)\text{-ARCS}) \leq w(d\text{-ARCS})$ for all $d \geq 2$ and all directed instances, and both inequalities are tight.

Proof. The inequality of Item 1 holds because we optimize over a larger set to obtain d-AF. It is tight, e.g., if all edge weights are equal.

The inequality of Item 2 holds since d-ARCS is Eulerian and connected. Thus, we can obtain a TSP tour by taking shortcuts and the triangle inequality guarantees that this does not increase the weight. It is tight, e.g., for the following instance: We have two clusters of $d + 1$ vertices each, and within each cluster, the weights are 0. Between the clusters, the weights are 1. Both ATSP and d-ARCS use only two edges between the two clusters, one in each direction. Thus, they have equal weight.

The first part of Item 3 is straightforward, because d-regular directed graphs stand in one-to-one correspondence to d-regular bipartite graphs, whose edges can be partitioned into d perfect matchings [19, Lemma 1.4.17]. Tightness holds if all edge weights are equal.

The tightness of the second inequality of Item 3 follows from the tightness of Item 2 and the fact that 1-ARCS = ATSP. Its proof is similar to the proof of Lemma 2.1(8): Let R be a minimum-weight connected d-factor. We iteratively decrease indegree and outdegree of every vertex by one. The invariant that we use is that the graph is strongly connected. This holds for every weakly connected directed graph where indegree equals outdegree at every vertex.

Consider a vertex v with indegree d and outdegree d. First, assume that v is a cut vertex, and let $C_1, \ldots, C_k$ be the components obtained by removing v. Since R is strongly connected, there is an $x \in C_1$ and $y \in C_2$ with $(x,v),(v,y) \in R$. We remove these two edges and add (x,y). The resulting graph is weakly connected and thus strongly connected.

If v is not a cut vertex, let y be arbitrary with $(v,y) \in R$. Since y has indegree at most d, there must be an x with $(x,v) \in R$ and $(x,y) \notin R$. We replace (x,v) and (v,y) by (x,y). Since v is no cut vertex, the resulting graph is connected. We iterate this process until we have a $(d-1)$-regular connected graph. By the triangle inequality, its weight is at most $w(d\text{-ARCS})$. □

3 Approximation Algorithms

3.1 3-Approximation for Min-d-RCS and Min-d-R2CS

The 3-approximation that we present in this section works for all d, odd or even. It also works for d growing as a function of n. An interesting feature of this algorithm, and possibly an indication that a better approximation ratio is possible for Min-d-RCS, is that the same algorithm provides an approximation ratio of 3 for both Min-d-RCS and Min-d-R2CS. In fact, we compute a 2-edge-connected d-regular graph that weighs at most three times the weight of the optimal connected d-regular graph.

First we make some preparatory observations on 2-edge-connectedness. Given a connected graph $G = (V, E)$, we can create a tree $T(G)$ as follows: We have a vertex for every maximal subgraph of G that is 2-edge-connected (called a 2-edge-connected component), and two such vertices are connected if the corresponding components are connected in G. In this case, they are connected by a bridge edge. Now consider a leaf of tree $T(G)$ and its corresponding 2-edge-connected component C. Since C is a leaf in $T(G)$, it is only incident to a single bridge edge e in G. Now assume that G is d-regular with $d \geq 3$ odd (for $d = 2$, any connected graph is also 2-edge-connected). Let u be the vertex of C that is incident to e. Then u must be incident to $d-1$ other vertices in C. Thus, C has at least d vertices. Since the $d-1$ neighbors of u are not incident to bridge edges, they must be adjacent to other vertices in C. Since G is d-regular, C has at least $d+1$ vertices and more than $d^2/2 > d$ edges. Therefore, there exists an edge e' in C that is not incident to u, i.e., e' does not share an endpoint with a bridge edge.

If G is not connected, we have exactly the same properties with "tree" replaced by "forest".

To simplify notation in the algorithm, let $k = k(G)$ denote the number of 2-edge-connected components of G that are leaves in the forest described above, and let $L_1(G), \ldots, L_k(G)$ denote the 2-edge-connected components of a graph G that correspond to leaves in the tree described above. For such an $L_i(G)$, let $e_i(G)$ denote an edge that is not adjacent to a bridge edge in G. The choice of $e_i(G)$ is arbitrary.

We prove that Algorithm 1 is a 3-approximation for both Min-d-RCS and Min-d-R2CS by a series of lemmas. Since the set of vertices is fixed, we sometimes identify graphs with their edge set. In particular, R denotes both the connected d-factor that we compute and its edge set.

> **input** : undirected complete graph $G = (V, E)$, edge weights w, $d \geq 2$
> **output**: 2-edge-connected d-factor R of G
> **1** compute a minimum-weight d-factor d-F of G;
> **2** $k \leftarrow k(d$-F$)$
> **3** $Q \leftarrow \{e_1, \ldots, e_k\}$ with $e_i = e_i(d$-F$) = \{u_i, v_i\}$
> **4** compute MST of G;
> **5** duplicate each edge of MST and take shortcuts to obtain a Hamiltonian cycle H
> **6** take shortcuts to obtain from H a Hamiltonian cycle H' through $\{u_1, \ldots, u_k\}$, assume
>   w.l.o.g. that H' traverses the vertices in the order $u_1, \ldots, u_k, u_1$
> **7** obtain R from d-F by adding the edges $\{u_i, v_{i+1}\}$ (with $k + 1 = 1$) and removing Q

Algorithm 1: 3-approximation for Min-d-RCS and Min-d-R2CS.

Lemma 3.1. *Assume that R is computed as in Algorithm 1. Then R is a d-regular spanning subgraph of G.*

Proof. If R does not contain multiple edges between the same pair of vertices, then R is d-regular since we obtain R from a d-factor, remove one edge incident to each u_i and v_i, and add one edge incident to each u_i and v_i. We now show that indeed we have that R does not contain multiple edges between the same pair of vertices. This can only happen if some edge $e = \{u_i, v_{i+1}\}$ is present in d-F. Since e connects two 2-edge-connected components, this can only happen if e is a bridge edge. This is not the case as none of the vertices u_i and v_i are incident to a bridge edge in d-F by the choice of the edges e_i in Line 3 of Algorithm 1. $\square$

Lemma 3.2. *Assume that R is computed as in Algorithm 1. Then R is 2-edge-connected.*

Proof. First, we observe that R is connected: We do not remove any bridge edges and we remove at most one edge per 2-edge-connected component of d-F. Furthermore, the 2-edge-connected components that are not already connected in d-F are connected via H'. To show that R is 2-edge-connected, we show that the removal of a single edge does not disconnect the graph.

If we remove an edge (u_i, v_{i+1}), then we still have a connection between u_i and v_{i+1} via $u_{i+1}, u_{i+2}, \ldots, u_{i-2}, u_{i-1}$. If we remove a bridge edge $e = (u, v)$ of d-F, then both u and v must each be connected to at least one 2-edge-connected component of d-F. Since those 2-edge-connected components are also connected through H', R remains to be connected: First, if we remove a non-bridge edge e within a 2-edge-connected component C of d-F, then this also does not disconnect the graph. If C does not correspond to a leaf in $T(d$-F$)$, then it is still 2-edge-connected in R. Thus removing one edge does not disconnect the graph. Second, if C is a leaf in $T(d$-F$)$, then an edge $e = (u, v)$ is removed from C in R. If removing another edge separates C in two components, then u and v must be in separate components, but then these components are still connected through H'. Thus, R remains to be connected also in this case. $\square$

Lemma 3.3. *Assume that R is computed as in Algorithm 1. Then $w(R) \leq 3 \cdot w(d$-RCS$) \leq 3 \cdot w(d$-R2CS$)$.*

Proof. The second inequality follows from Lemma 2.1(1). For the first inequality, we observe that $w(\mathsf{MST}) \leq w(d$-RCS$)$ (Lemma 2.1(1)) and $w(d$-F$) \leq w(d$-RCS$)$. Also, $w(\{u_i, v_{i+1}\}) \leq$

$w(\{u_i, u_{i+1}\}) + w(\{u_{i+1}, v_{i+1}\})$ by the triangle inequality. We have

$$w(R) \leq \underbrace{w(H') + w(Q)}_{\text{adding the } \{u_i, v_{i+1}\}\text{'s}} - \underbrace{w(Q)}_{\text{removing } Q} + w(d\text{-F})$$

$$\leq w(H) + w(d\text{-F}) \leq 2 \cdot w(\mathsf{MST}) + w(d\text{-F}) \leq 3 \cdot w(d\text{-RCS}).$$

The second-to-last inequality holds since we can obtain a TSP tour by duplicating all edges of an MST and taking shortcuts. □

The following theorem is an immediate consequence of the lemmas above.

Theorem 3.4. *For all d, Algorithm 1 is a polynomial-time 3-approximation for* Min-d-RCS *and* Min-d-R2CS*. This includes the case that d is a function of n.*

Remark 3.5. *If we are only interested in a 3-approximation for* Min-d-RCS *and not for* Min-d-R2CS*, then we can simplify Algorithm 1 a bit: we only pick one non-bridge edge for each component and not for every 2-edge-connected component. The rest of the algorithm and its analysis remain the same. However, this does not seem to improve the worst-case approximation ratio.*

Remark 3.6. *The analysis is tight in the following sense: By Lemma 2.1(3), a minimum-weight 2-edge-connected d-factor can be three times as heavy as a minimum-weight connected d-factor. Thus, any algorithm that outputs a 2-edge-connected d-factor cannot achieve an approximation ratio better than 3. Furthermore, since $w(\mathsf{MST}) \leq w(d\text{-R2CS})$ and $w(d\text{-F}) \leq w(d\text{-R2CS})$ are tight (Lemma 2.1(1) and (2)), the analysis is essentially tight. If we only require connectedness and not 2-edge-connectedness, we see that the analysis cannot be improved since $w(\mathsf{TSP}) \leq 2w(d\text{-RCS})$ and $w(d\text{-F}) \leq w(d\text{-RCS})$ are tight.*

However, it is reasonable to assume that not all these inequalities can be tight at the same time and, in addition, shortcutting of the duplicated MST to obtain a TSP tour through $u_1, \ldots, u_k$ does not yield an improvement. Therefore, it might be possible to improve the analysis and show that Algorithm 1 achieves a better approximation ratio than 3.

Remark 3.7. *Lines 4 and 5 of Algorithm 1 are in fact simply the double-tree heuristic for* Min-TSP *[29, Section 2.4]. One might be tempted to construct a better tour using Christofides' algorithm [29, Section 2.4], which achieves a ratio of 3/2 instead of only 2. However, in the analysis we compare the optimal solution for* Min-d-RCS *to the MST, and we know that $w(\mathsf{MST}) \leq w(d\text{-RCS}) \leq w(d\text{-R2CS})$. If we use Christofides' algorithm directly, we have to compare a TSP tour to the minimum-weight connected d-factor. In particular for odd d, we have that for some instances $w(\mathsf{TSP}) \geq (\frac{4}{3} - o(1)) \cdot w(d\text{-R2CS}) \geq (\frac{4}{3} - o(1)) \cdot w(d\text{-RCS})$ (Lemma 2.1(7)). Even if this is the true bound – as it is for $d = 3$ (Lemma 2.1(6)) –, the TSP tour constructed contributes with a factor 3/2 times 4/3, which equals 2, to the approximation ratio, which is no improvement.*

3.2 $(r+1)$-Approximation

In this section, we give an $(r+1)$-approximation for Min-d-RCS for even values of d and Min-d-ARCS for all values of d. Here, r is the ratio within which Min-TSP (for Min-d-RCS) or Min-ATSP (for Min-d-ARCS) can be approximated. This means that we currently have $r = 3/2$ for the symmetric case by Christofides' algorithm [29, Section 2.4] and, for the asymmetric

---

> **input** : undirected or directed complete graph $G = (V, E)$, edge weights w, d
> **output**: connected d-factor R of G
> **1** compute a minimum-weight d-factor C of G
> **2** let $C_1, \dots, C_k$ be the connected components of C, and let $e_i = (u_i, v_i)$ be any edge of C_i
> **3** compute a TSP tour H using an approximation algorithm with ratio r
> **4** take shortcuts to obtain from H a TSP tour H' through $\{u_1, \dots, u_k\}$, assume w.l.o.g.
> that H' traverses the vertices in the order $u_1, \dots, u_k, u_1$
> **5** obtain R from C by adding the edges (u_i, v_{i+1}) (with $k+1 = 1$) and removing $e_1, \dots, e_k$

Algorithm 2: $(r + 1)$-approximation for Min-d-RCS for even d and Min-d-ARCS.

---

case, we have either $r = O(\log n / \log \log n)$ if we use the randomized algorithm by Asadpour et al. [1] or $r = \frac{2}{3} \cdot \log_2 n$ if we use Feige and Singh's deterministic algorithm [10]. Although the algorithm is a simple modification of Algorithm 1, we summarize it as Algorithm 2 for completeness.

Theorem 3.8. *If* Min-TSP *can be approximated in polynomial time within a factor of r, then Algorithm 2 is a polynomial-time $(r + 1)$-approximation for* Min-d-RCS *for all even d.*

If Min-ATSP *can be approximated in polynomial time within a factor of r, then Algorithm 2 is a polynomial-time $(r + 1)$-approximation for* Min-d-ARCS *for all d.*

The results still hold if d is part of the input.

Proof. Let T be an optimal TSP tour, and let O be an optimal connected d-factor. Let C be a minimum-weight d-factor, as computed by Algorithm 2. We have $T = \mathsf{ATSP}$, $O = d\text{-}\mathsf{ARCS}$, and $C = d\text{-}\mathsf{AF}$ if the input graph is asymmetric and $T = \mathsf{TSP}$, $O = d\text{-}\mathsf{RCS}$, and $C = d\text{-}\mathsf{F}$ if the input graph is symmetric.

By Lemma 2.1 and Lemma 2.2, we have $w(T) \leq w(O)$ and $w(C) \leq w(O)$. Removing and adding edges as in Line 5 of Algorithm 2 yields again a d-factor. For the asymmetric case, any component is strongly connected. After removal of one edge per component, it is still weakly connected. For the symmetric case, any component is 2-edge-connected. Thus, the removal of edges in Line 5 does not split any component. Hence, the addition of edges in Line 5 yields a connected d-factor R. By the triangle inequality, we have $w(R) \leq w(C) + w(H') \leq w(C) + w(H)$. Since we use an r-approximation to obtain H, we thus have $w(R) \leq w(C) + rw(T) \leq (r + 1) \cdot w(O)$. □

4 Hardness Results

4.1 TSP-Inapproximability

In this section, we prove that Min-d-RCS cannot be approximated better than Min-TSP.

Theorem 4.1. *For every $d \geq 2$, if* Min-d-RCS *can be approximated in polynomial time within a factor of r, then* Min-TSP *can be approximated in polynomial time within a factor of r.*

Proof. We show that Min-d-RCS can be used to approximate Min-TSP. Let the instance of Min-TSP be given by a complete graph $G = (V, E)$ and edge weights $w = (w_e)_{e \in E}$ that satisfy the triangle inequality. Let $n = |V|$. We construct an instance of Min-d-RCS as follows: The instance consists of a complete graph $H = (V', E')$. Here $V' = \bigcup_{v \in V} V_v$, where $V_v = \{v_1, v_2, \dots, v_{d+1}\}$, i.e., H contains $(d+1) \cdot n$ vertices. We assign edge weights $\tilde{w}$ as follows:

- $\tilde{w}_{\{v_i,v_j\}} = 0$ for all $v \in V$, $i \neq j$,

- $\tilde{w}_{\{u_i,v_j\}} = w_{\{u,v\}}$ for all $u \neq v$, i and j.

Every TSP tour T of G maps to a connected d-factor R of H of the same weight: We give T an orientation. For an edge from u to v in T, we include $\{u_1, v_2\}$ in R. Adding all edges except $\{v_1, v_2\}$ to R within each V_v yields a connected d-factor R. Clearly, $\tilde{w}(F) = w(T)$.

Now assume that we have a connected d-factor R of H. We claim that we can construct a TSP tour T of G with $w(T) \leq \tilde{w}(R)$. We construct a multiset T' of edges of G as follows: For each edge $\{u_i, v_j\}$ of R, if $u \neq v$, we add an edge $\{u, v\}$ to T'. Otherwise, if $u = v$, we ignore the edge. The sum of the degrees in R of all vertices in each set V_v is equal to $(d+1)d$ and is therefore even. Thus, for each v, the number of edges leaving V_v in R, which equals the number of edges incident to v in T' by construction, is even as well. Since R is connected, the multigraph $G' = (V, T')$ is connected as well. By construction, $w(T') = \tilde{w}(R)$. Since G' is connected and all its vertices have even degree, G' is Eulerian. Therefore, we can obtain a TSP tour T from T' by taking shortcuts. By the triangle inequality, $w(T) \leq w(T') = \tilde{w}(R)$. $\qquad\square$

The same construction as in the proof of Theorem 4.1 yields the same result for Min-d-R2CS. A similar construction yields the same result for Min-d-ARCS.

Corollary 4.2. *For every $d \geq 2$, if* Min-d-R2CS *can be approximated in polynomial time within a factor of r, then* Min-TSP *can be approximated in polynomial time within a factor of r.*

Corollary 4.3. *For every $d \geq 2$, if* Min-d-ARCS *can be approximated in polynomial time within a factor of r, then* Min-ATSP *can be approximated in polynomial time within a factor of r.*

Min-TSP, Min-ATSP, Max-TSP, and Max-ATSP are APX-hard [23]. Furthermore, the reduction from Min-TSP to Min-d-RCS is in fact an L-reduction [22] (see also Shmoys and Williamson [29, Section 16.2]). This proves the APX-hardness of Min-d-RCS for all d. The reductions from Min-TSP to Min-d-R2CS and from Min-ATSP to Min-d-ARCS work in the same way. Furthermore, by reducing from Max-TSP and Max-ATSP in a similar way (here, the edges between the copies of a vertex have high weight), we obtain APX-hardness for Max-d-RCS and Max-d-ARCS as well.

Corollary 4.4. *For every fixed $d \geq 2$, the problems* Min-d-RCS, Min-d-R2CS, *and* Max-d-RCS *are* APX-*complete. For every fixed $d \geq 1$,* Min-d-ARCS *and* Max-d-ARCS *are* APX-*complete.*

4.2 Hardness for Growing d

In this section, we generalize the NP-hardness proof for d-RCS by Cheah and Corneil [6] to the case that d grows with n. Furthermore, we extend Theorem 4.1 and Corollaries 4.2 and 4.3 and the APX-hardness of the minimization variants (Corollary 4.4) to growing d. The APX-hardness of Max-d-RCS and Max-d-ARCS does not transfer to growing d – both can be approximated within a factor of $1 - O(1/d^2)$, which is $1 - o(1)$ for growing d.

Let us consider Cheah and Corneil's [6, Section 3.2] reduction from 2-RCS, i.e., the Hamiltonian cycle problem, to d-RCS. Crucial for their reduction is the notion of the d-expansion of a vertex v, which is obtained as follows:

1. We construct a gadget G_{d+1} by removing a matching of size $\lceil \frac{d}{2} \rceil - 1$ from a complete graph on $d + 1$ vertices.

2. We connect each vertex whose degree has been decreased by one to v.

The reduction itself takes a graph G for which we want to test if $G \in$ 2-RCS and maps it to a graph $R_d(G)$ as follows: For even d, $R_d(G)$ is the graph obtained by performing a d-expansion for every vertex of G. For odd d, the graph $R_d(G)$ is obtained by doing the following for each vertex v of G: add vertices $u_1, u_2, \ldots, u_{d-2}$; connect v to $u_1, \ldots, u_{d-2}$; perform a d-expansion on $u_1, \ldots, u_{d-2}$. We have $G \in$ 2-RCS if and only if $R_d(G) \in d$-RCS.

We note that $R_d(G)$ has $(d+2) \cdot n$ vertices for even d and $\Theta(d^2 n)$ vertices for odd d and can easily be constructed in polynomial time since $d < n$.

Theorem 4.5. *For every fixed $\varepsilon > 0$, there is a function $f = \Theta(n^{1-\varepsilon})$ that maps to even integers such that f-RCS is NP-hard.*

For every fixed $\varepsilon > 0$, there is a function $f = \Theta(n^{\frac{1}{2}-\varepsilon})$ that maps to odd integers such that f-RCS is NP-hard.

Proof. We first present the proof for the case that we map to even integers. After that, we briefly point out the difference for odd integers.

We choose $d = 2\lceil n^{\frac{1-\varepsilon}{\varepsilon}} \rceil$ and apply $R = R_d(G)$. The graph R has $g(n) = n \cdot (2\lceil n^{\frac{1-\varepsilon}{\varepsilon}} \rceil + 2)$ vertices since d is even. We have $g = \Theta(n^{1/\varepsilon})$. Now we determine f: we require $f(g(n)) = d = 2\lceil n^{\frac{1-\varepsilon}{\varepsilon}} \rceil$. This can be achieved because $g = \omega(n)$ is an injective function.

Expressed as a function of g, we have $d = \Theta(g(n)^{1-\varepsilon})$. For natural numbers that are not images of g, we interpolate f to maintain the growth bound. Thus, $f(n) = \Theta(n^{1-\varepsilon})$.

Let us now point out the differences for functions f mapping to odd integers. In this case, since the reduction for d maps to graphs of size $\Theta(d^2 n)$, we have to choose $d = \Theta(n^{\frac{1-\varepsilon}{2\varepsilon-1}})$. This, however, works only up to $\varepsilon > 1/2$ or functions up to $n^{\frac{1}{2}-\varepsilon}$. □

In the same way as the NP-completeness, the inapproximability can be transferred. The reduction creates graphs of size $(d+1) \cdot n$. The construction is the same as in Section 4.1, and the proof follows the line of the proof of Theorem 4.5. Here, however, we do not have to distinguish between odd and even d for the symmetric variant, as the reduction in Section 4.1 is the same for both cases.

Theorem 4.6. *For every fixed $\varepsilon > 0$, there is a function $f = \Theta(n^{1-\varepsilon})$ such that Min-f-RCS and Min-f-R2CS are APX-hard and cannot be approximated better than Min-TSP.*

For every fixed $\varepsilon > 0$, there is a function $f = \Theta(n^{1-\varepsilon})$ such that Min-f-ARCS is APX-hard and cannot be approximated better than Min-ATSP.

5 Further Algorithms

5.1 2-Approximation for $d \geq n/3$

If $d \geq n/3$, then we easily get a better approximation algorithm for Min-d-R2CS and Min-d-RCS. In this case, d-F consists either of a single component – then we are done – or of two components C_1 and C_2 with $C_i = (V_i, E_i)$. In the latter case, we proceed as follows: first, find the lightest edge $e = \{u, v\}$ with $u \in V_1$ and $v \in V_2$. Second, choose any edges $\{u, u'\} \in E_1$ and $\{v, v'\} \in E_2$. Third, remove $\{u, u'\}$ and $\{v, v'\}$ and add $\{u, v\}$ and $\{u', v'\}$. The increase in weight is at most $2 \cdot w(\{u, v\})$ by the triangle inequality.

The resulting graph is clearly d-regular. It is connected since C_1 and C_2 are 2-edge-connected: they both consist of at most $\frac{2n}{3} - 1$ vertices and are d-regular with $d \geq n/3$. Thus, they are even

Hamiltonian by Dirac's theorem [28]. Furthermore, any connected d-regular graph must have at least two edges connecting V_1 and V_2: If d is even, then this follows by 2-edge-connectedness. If d is odd, then $|V_1|$ and $|V_2|$ are even and, thus, an even number of edges must leave either of them. Thus, $w(\{u, v\}) \leq \frac{1}{2} \cdot w(d\text{-RCS})$. Since we add at most $2 \cdot w(\{u, v\})$ and also have $w(d\text{-F}) \leq w(d\text{-RCS})$, we obtain the following theorem.

Theorem 5.1. *For $d \geq n/3$, there is a polynomial-time 2-approximation for* Min-d-RCS*.*

5.2 Decision Problem for $d = \lceil \frac{n}{2} \rceil - 1$

For $d \geq n/2$, any d-factor is immediately connected and also the minimization variant can be solved efficiently. In this section, we slightly extend this to the case of $d \geq \frac{n}{2} - 1$.

We assume that the input graph G is connected. To show that the case $d = \lceil \frac{n}{2} \rceil - 1$ is in P, we compute a d-factor. If none exists or we obtain a connected d-factor, then we are done. Otherwise, we have a d-factor consisting of two components C_1 and C_2 which are both cliques of size $n/2$. If G contains a cut vertex, say, $u \in C_1$, then this is the only vertex with neighbors in C_2. In this case, G does not contain a connected d-factor. If G does not contain a cut vertex, there are two disjoint edges $e = \{u, v\}$, $e' = \{u', v'\}$ with $u, u' \in C_1$ and $v, v' \in C_2$. Adding e and e' and removing $\{u, u'\}$ and $\{v, v'\}$ yields a connected d-factor.

Theorem 5.2. d-RCS *is in* P *for every d with $d \geq \frac{n}{2} - 1$.*

5.3 Approximating Max-d-ARCS

The approximation algorithm for Max-d-RCS [2] can easily be adapted to work for Max-d-ARCS: We compute a directed d-factor of maximum weight. Any component consists of at least $d + 1$ vertices, thus at least $d \cdot (d+1)$ arcs. We remove the lightest arc of every component and connect the resulting (still at least weakly connected) components arbitrarily to obtain a connected d-factor. Since we have removed at most a $\frac{1}{d \cdot (d+1)}$-fraction of the weight, we obtain the following result.

Theorem 5.3. *For every d,* Max-d-ARCS *can be approximated within a factor of $1 - \frac{1}{d \cdot (d+1)}$.*

6 Open Problems

An obvious open problem is to improve the approximation ratios. Apart from this, let us mention two open problems: First, is it possible to achieve constant factor approximations for minimum-weight k-edge-connected or k-vertex-connected d-regular graphs? Without the regularity requirement, the problem of computing minimum-weight k-edge-connected graphs can be approximated within a factor of 2 [18] and the problem of computing minimum-weight k-vertex-connected graphs can be approximated within a factor of $2 + 2 \cdot \frac{k-1}{n}$ for metric instances [16] and still within a factor of $O(\log k)$ if the instances are not required to satisfy the triangle inequality [7]. We refer to Khuller and Raghavachari [17] for a concise survey.

Second, we have seen that $(\lceil \frac{n}{2} \rceil - 1)$-RCS $\in$ P, but we do not know if Min-$(\lceil \frac{n}{2} \rceil - 1)$-RCS can be solved in polynomial time as well. In addition, we conjecture that also $(\lceil \frac{n}{2} \rceil - k)$-RCS is in P for any constant k.

References

[1] Arash Asadpour, Michel X. Goemans, Aleksander Madry, Shayan Oveis Gharan, and Amin Saberi. An $O(\log n/\log \log n)$-approximation algorithm for the asymmetric traveling salesman problem. In *Proc. of the 21st Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 379–389. SIAM, 2010.

[2] Alexey E. Baburin and Edward Kh. Gimadi. Approximation algorithms for finding a maximum-weight spanning connected subgraph with given vertex degrees. In *Operations Research Proceedings 2004*, pages 343–351, 2005.

[3] Alexey E. Baburin and Edward Kh. Gimadi. Polynomial algorithms for some hard problems of finding connected spanning subgraphs of extreme total edge weight. In *Operations Research Proceedings 2006*, pages 215–220, 2007.

[4] Alexey E. Baburin and Edward Kh. Gimadi. An approximation algorithm for finding a d-regular spanning connected subgraph of maximum weight in a complete graph with random weights of edges. *Journal of Applied and Industrial Mathematics*, 2(2):155–166, 2008.

[5] Yuk Hei Chan, Wai Shing Fung, Lap Chi Lau, and Chun Kong Yung. Degree bounded network design with metric costs. *SIAM Journal on Computing*, 40(4):953–980, 2011.

[6] F. Cheah and Derek G. Corneil. The complexity of regular subgraph recognition. *Discrete Applied Mathematics*, 27(1-2):59–68, 1990.

[7] Joseph Cheriyan, Santosh Vempala, and Adrian Vetta. An approximation algorithm for the minimum-cost k-vertex connected subgraph. *SIAM Journal on Computing*, 32(4):1050–1055, 2003.

[8] Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards*, 69B(1–2):125–130, 1965.

[9] Bruno Escoffier, Laurent Gourvès, and Jérôme Monnot. Complexity and approximation results for the connected vertex cover problem in graphs and hypergraphs. *Journal of Discrete Algorithms*, 8(1):36–49, 2010.

[10] Uriel Feige and Mohit Singh. Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In *Proc. of the 10th Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 4627 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2007.

[11] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximation for finding a maximum weight Hamiltonian cycle. *Operations Research*, 27(4):799–809, 1979.

[12] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[13] Edward Kh. Gimadi and A. I. Serdyukov. A problem of finding the maximal spanning connected subgraph with given vertex degrees. In *Operations Reserach Proceedings 2000*, pages 55–59. Springer, 2001.

[14] Sudipto Guha and Samir Khuller. Improved methods for approximating node weighted steiner trees and connected dominating sets. *Information and Computation*, 150(1):57–74, 1999.

[15] Haim Kaplan, Moshe Lewenstein, Nira Shafrir, and Maxim I. Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *Journal of the ACM*, 52(4):602–626, 2005.

[16] Samir Khuller and Balaji Raghavachari. Improved approximation algorithms for uniform connectivity problems. *Journal of Algorithms*, 21(2):434–450, 1996.

[17] Samir Khuller and Balaji Raghavachari. Graph connectivity. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*. Springer, 2008.

[18] Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *Journal of the ACM*, 41(2):214–235, 1994.

[19] László Lovász and Michael D. Plummer. *Matching Theory*, volume 121 of *North-Holland Mathematics Studies*. Elsevier, 1986.

[20] Dáaniel Marx, Barry O'sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30:1–30:35, 2013.

[21] Katarzyna Paluch, Marcin Mucha, and Aleksander Madry. A 7/9 approximation algorithm for the maximum traveling salesman problem. In *Proc. of the 12th Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 5687 of *Lecture Notes in Computer Science*, pages 298–311. Springer, 2009.

[22] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[23] Christos H. Papadimitriou and Mihalis Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.

[24] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

[25] Paul D. Seymour. On multi-colourings of cubic graphs, and conjectures of Fulkerson and Tutte. *Proceedings of the London Mathematical Society*, s3-38(3):423–460, 1979.

[26] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proc. of the 39th Ann. Int. Symp. on Theory of Computing (STOC)*, pages 661–670. ACM, 2007.

[27] William T. Tutte. A short proof of the factor theorem for finite graphs. *Canadian Journal of Mathematics*, 6:347–352, 1954.

[28] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.

[29] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.