IEICE TRANS. INF. & SYST., VOL.E101–D, NO.3 MARCH 2018

PAPER Special Section on Foundations of Computer Science — Frontiers of Theoretical Computer Science —

Complexity of the Minimum Single Dominating Cycle Problem for Graph Classes

Hiroshi ETO[†a)], **Hiroyuki KAWAHARA**[††b)], ***Nonmembers***, **Eiji MIYANO**[††c)], ***Member***, ***and*** **Natsuki NONOUE**[††d)], ***Nonmember***

SUMMARY In this paper, we study a variant of the Minimum Dominating Set problem. Given an unweighted undirected graph $G = (V, E)$ of $n = |V|$ vertices, the goal of the Minimum Single Dominating Cycle problem (MinSDC) is to find a single shortest cycle which dominates all vertices, i.e., a cycle C such that for the set $V(C)$ of vertices in C and the set $N(V(C))$ of neighbor vertices of C, $V(G) = V(C) \cup N(V(C))$ and $|V(C)|$ is minimum over all dominating cycles in G [6], [17], [24]. In this paper we consider the (in)approximability of MinSDC if input graphs are restricted to some special classes of graphs. We first show that MinSDC is still $\mathcal{NP}$-hard to approximate even when restricted to planar, bipartite, chordal, or r-regular ($r \geq 3$). Then, we show the $(\ln n + 1)$-approximability and the $(1 - \varepsilon)\ln n$-inapproximability of MinSDC on split graphs under $\mathcal{P} \neq \mathcal{NP}$. Furthermore, we explicitly design a linear-time algorithm to solve MinSDC for graphs with bounded treewidth and estimate the hidden constant factor of its running time-bound.

key words: *minimum single dominating problem, graph classes, (in)tractability, (in)approximability*

1. Introduction

Let $G = (V(G), E(G))$ be a simple, undirected, unweighted and connected graph, where $V(G)$ and $E(G)$ denote the set of vertices and the set of edges, respectively. For a vertex v, let $N(v)$ be the neighbors of v in G which does not include v itself. Also, for a set $S \subseteq V(G)$, let $N(S)$ be the neighbors of S which does not include S itself. Given a graph $G = (V(G), E(G))$, a set $D \subseteq V(G)$ of vertices is called a *dominating set* in the graph G if $V(G) = D \cup N(D)$, i.e., every vertex $v \in V(G) - D$ must be adjacent at least to one vertex in D. The problem of finding the *minimum cardinality* dominating set in the input graph is known as the Minimum Dominating Set problem (MinDS), which is one of the central problems in graph theory, operations research, and computational geometry; many different problems in diverse fields have been modeled using dominating sets. For example, the *Art Gallery* or *Museum* problem, which is a well-studied visibility problem in computational geometry, is naturally formulated as MinDS on the *visibility graph* of the polygon, i.e., several (static) watchmen must be placed on vertices and all the vertices must be guarded by the watchmen who look out for their vertices and neighbor vertices (e.g., see [4], [22]). Furthermore, many different variants of the dominating set problem have been introduced such as the Minimum Dominating Independent Set and Minimum Dominating Connected Set problems (e.g., see [13]).

In this paper, we study another variant of MinDS, motivated by the Watchman Route problem [3], which is one of the famous path planning problems in computational geometry and robotics. The goal of the Watchman Route problem is to find a shortest route R such that a moving watchman follows R from a point s back to itself with property that each point in a given space is visible from at least one point along the route R. Our problem is named Minimum Single Dominating Cycle problem (MinSDC for short): Given a graph $G = (V, E)$, the goal of MinSDC is to find a *single* shortest cycle which dominates all vertices, i.e., the connected simple cycle C such that for the set $V(C)$ of vertices in C and the set $N(V(C))$ of neighbor vertices of C, $V(G) = V(C) \cup N(V(C))$ and $|V(C)|$ is minimum over all dominating cycles in G [6], [17], [24]. It is easy to see that MinSDC is $\mathcal{NP}$-hard in general since it can be seen a "merged" problem of MinDS and the Hamiltonian Cycle problem (HC), which is also one of the well known $\mathcal{NP}$-hard ones [13]. Furthermore, unfortunately, Proskurowski and Sysło [24], and Colbourn and Stewart [6] prove that MinSDC remains $\mathcal{NP}$-hard even if the input graph is either planar, bipartite or split. On the other hand, fortunately, it is known that MinSDC becomes tractable if the input graphs are restricted to 2-tree [23], two-connected outerplanar [24], series-parallel [6], circular-arc graphs [17], and so on.

In this paper, we focus on the approximability/inapproximability and/or the tractability/intractability of MinSDC on subclasses of graphs, including planar, chordal, split, bipartite, regular graphs, and graphs with bounded treewidth. As far as the authors know, this is the first paper which explicitly investigates the (in)approximability of MinSDC. The following is a list of our main results shown in this paper, the tractability and the (in)approximability for the graph classes: (i) MinSDC is $\mathcal{NP}$-hard to approximate even if the input graph is either planar, bipartite, chordal, chordal bipartite, or r-regular ($r \geq 3$). (ii) MinSDC can be approximated within a factor of $(\ln n+1)$ if the input is a split graph with n vertices. (iii) The $(\ln n+1)$-approximability for MinSDC on split graphs is the best possible; it is $\mathcal{NP}$-hard

Manuscript received March 24, 2017.
Manuscript revised July 8, 2017.
Manuscript publicized December 19, 2017.

[†]The author is with Kyushu University, Fukuoka-shi, 812–8581 Japan.
[††]The authors are with Kyushu Insitute of Technology, Iizuka-shi, 820–8502 Japan.
a) E-mail: h-eto@econ.kyushu-u.ac.jp
b) E-mail: kawahara_h@theory.ces.kyutech.ac.jp
c) E-mail: miyano@ces.kyutech.ac.jp
d) E-mail: nonoue@theory.ces.kyutech.ac.jp
DOI: 10.1587/transinf.2017FCP0007

to approximate MinSDC to within a factor of $(1-\varepsilon)\ln n$ for every $\varepsilon > 0$ for split graphs of n vertices. (iv) We explicitly design a dynamic-programming algorithm which solves MinSDC in linear time for graphs with bounded treewidth.

It is known [7] that any optimization problem definable in monadic second order logic can be solved in linear time for graphs with bounded treewidth. Indeed (the decision version of) MinSDC can be expressed in monadic second order logic as shown in Sect. 5 later. However, the algorithm obtained by this method is hard to implement and to estimate its running time; it is generally very slow since the dependence on treewidth in the hidden constant factor of the running time is a tower of exponentials. On the other hand, our algorithm is simple and thus the hidden constant factor of its running time can be obtained.

In Sect. 2, we first give the formal definitions and the relationships of graph classes, some previous results, and then the inapproximability of the problem for planar, bipartite, chordal, chordal bipartite graphs. Sections 3 and 4 show the (in)approximability for regular and split graphs, respectively. The results for graphs with bounded treewidth are presented in Sect. 5.

2. Preliminaries

Problems and previous results. Let $G = (V, E)$ be an unweighted graph and let $n = |V|$. A graph S is a subgraph of G if $V(S) \subseteq V$ and $E(S) \subseteq E$. For a subset of vertices $U \subseteq V$, let $G[U]$ be the subgraph of G induced by U. For a subgraph S of G, if $E(S) = V(S) \times V(S)$, then S (or $G[V(S)]$) and $V(S)$ are called a *clique* and a *clique set*, respectively. A path of length ℓ, from a vertex v_0 to a vertex v_ℓ, is represented as a sequence of vertices $P_{v_0,v_\ell} = \langle v_0, v_1, \cdots, v_\ell\rangle$, or equivalently, it is often represented as a sequence of edges $P_{v_0,v_\ell} = \langle\{v_0, v_1\}, \{v_1, v_2\}, \cdots, \{v_{\ell-1}, v_\ell\}\rangle$. A cycle C is similarly written as $C = \langle v_0, v_1, \cdots, v_\ell, v_0\rangle$, or $C = \langle\{v_0, v_1\}, \{v_1, v_2\}, \cdots, \{v_\ell, v_0\}\rangle$. In this paper we deal with simple paths and simple cycles only. For a graph G and its vertex v, let $N(G, v) = \{u \in V(G) \mid (v, u) \in E(G)\}$, that is, the neighbors of v in G which does not include v itself. We denote by $deg(G, v) = |N(G, v)|$ the *degree* of v in G. If $u \notin G$, then we define $deg(G, u) \equiv 0$.

The definitions of graph classes are from [2]: A *chord* of a cycle is an edge between two vertices of the cycle that is not an edge of the cycle. A graph G is a *chordal graph* if each cycle in G of length at least four has at least one chord. A graph $G = (V, E)$ is a *split graph* if there is a partition of V into a clique set V_1 and an independent set V_2 such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$. Note that the class of split graphs is a subclass of chordal graphs. A bipartite graph is *chordal bipartite* if it has no induced cycle of length at least six. The definition of *treewidth* will be given in Sect. 5.

Our problem, Minimum Single Dominating Cycle (MinSDC), is formulated as the following minimization problem: Given a graph $G = (V, E)$, the objective of MinSDC is to find a single shortest cycle C which dominates all vertices. For MinSDC, an algorithm ALG is called σ-approximation algorithm and ALG's approximation ratio is σ if $ALG(G)/OPT(G) \leq \sigma$ holds for every input graph G, where $ALG(G)$ and $OPT(G)$ are the numbers of vertices of obtained dominating cycles by ALG and an optimal algorithm OPT, respectively. If G has no dominating cycle, $OPT(G)$ is defined to be zero.

The following decision version of MinSDC, we call SDC(ℓ), is previously considered in the literature [6], [24]: Given a graph $G = (V, E)$ and an integer ℓ, SDC(ℓ) determines whether the graph contains a single cycle of length ℓ or less which dominates all vertices. Actually, Colbourn and Stewart claim the $\mathcal{NP}$-completeness of SDC(ℓ) for several graph subclasses in [6] by providing a polynomial-time reduction from the Hamiltonian Cycle problem (HC), which determines whether a given undirected graph G contains at least one Hamiltonian cycle or not. Here we give a brief sketch of their proof: Consider the reduction such that given a graph G in a graph class $\mathcal{G}$, we adds a single vertex of degree one to every vertex $v \in V(G)$. We denote this reduction by α, and thus let $\alpha(G)$ be the resulting graph from G. Then, the following lemma is shown in [6].

Lemma 1 ([6])**.** Let $\mathcal{G}$ be a graph class for which HC is $\mathcal{NP}$-complete. If, for every $G \in \mathcal{G}$, $\alpha(G) \in \mathcal{G}$ holds, then SDC(ℓ) is $\mathcal{NP}$-complete for the graph class $\mathcal{G}$.

One can see that if a graph G is planar (bipartite, chordal, and chordal bipartite, resp.), then the graph $\alpha(G)$ reduced by α remains planar (bipartite, chordal, and chordal bipartite, resp.). Also, it is known [11], [14], [18], [20], [21] that HC on those graph classes remains $\mathcal{NP}$-complete.

Proposition 1 ([11], [14], [18], [20], [21])**.** SDC(ℓ) (and thus MinSDC) on planar, bipartite, chordal, and chordal bipartite graphs is $\mathcal{NP}$-hard.

On the other hand, MinSDC admits polynomial-time algorithms for the following graph classes:

Proposition 2 ([6], [17], [23], [24])**.** SDC(ℓ) (and thus MinSDC) is solvable in polynomial time for 2-tree, two-connected outerplanar, series-parallel, interval, and circular-arc graphs.

Inapproximability of MinSDC on planar, bipartite, chordal, and chordal bipartite graphs. Now we prove the inapproximability of MinSDC on graph subclasses. To do so, we consider another decision variant of MinSDC, called One Single Dominating Cycle (OneSDC): Given a graph $G = (V, E)$, OneSDC determines whether the graph contains a single cycle which dominates all vertices. Note that OneSDC simply asks for the existence of a single dominating cycle in G, and hence OneSDC is another decision version of MinSDC in the sense that the problem determines whether $OPT(G) > 0$ holds or not for a given graph G. We can show the following lemma by considering the reduction α provided in [6] from HC to OneSDC:

Lemma 2. Let $\mathcal{G}$ be a graph class for which HC is $\mathcal{NP}$-complete. If, for every $G \in \mathcal{G}$, $\alpha(G) \in \mathcal{G}$ holds, then

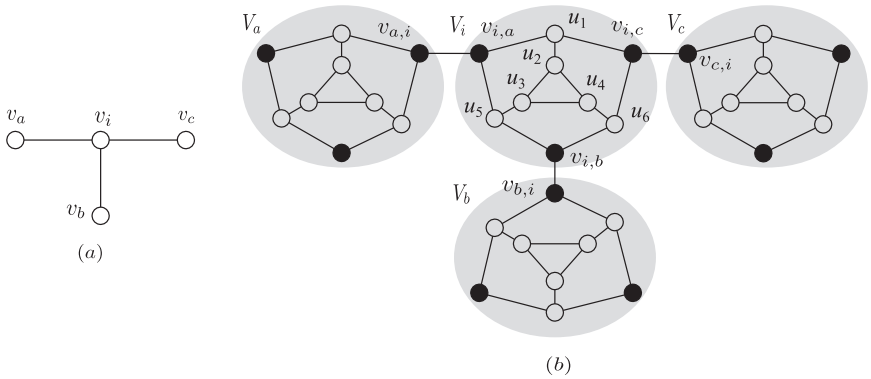


Fig. 1 (a) Input graph G of HC, (b) the corresponding graph H of OneSDC.

OneSDC is $\mathcal{NP}$-complete for the graph class $\mathcal{G}$.

Thus, we have:

Theorem 1. OneSDC on planar, bipartite, chordal, and chordal bipartite graphs is $\mathcal{NP}$-complete.

Theorem 1 implies the following inapproximability:

Corollary 1. Let $\rho(n) \geq 1$ be any polynomial-time computable function. For planar, bipartite, chordal, and chordal bipartite graphs, MinSDC admits no polynomial-time approximation algorithm with a factor of $\rho(n)$ unless $\mathcal{P} = \mathcal{NP}$.

Proof. Suppose for a contradiction that there exists a polynomial-time $\rho(n)$-approximation algorithm ALG for some polynomial-time computable function $\rho(n) > 1$ for MinSDC. Then, ALG can find a single dominating cycle in a given graph G in polynomial time such that the objective value $ALG(G)$ (i.e., length of the dominating cycle) satisfies $OPT(G) \leq ALG(G) \leq \rho \cdot OPT(G)$. Therefore, one can distinguish either $OPT(G) > 0$ or $OPT(G) = 0$ in polynomial time using ALG which admits the approximation ratio of $\rho(n)$. This is a contradiction unless $\mathcal{P} = \mathcal{NP}$, because Theorem 1 implies that it is $\mathcal{NP}$-complete to determine whether $OPT(G) > 0$ or not, for planar, bipartite, chordal or chordal bipartite graphs. □

3. Regular Graphs

In this section, we show the inapproximability of MinSDC on regular graphs, again via the $\mathcal{NP}$-completeness proof of OneSDC. It is important to note that if the reduction α in Sect. 2 reduces a regular graph G_R to $\alpha(G_R)$, then $\alpha(G_R)$ is not regular since the degree of the new vertices is one, but the degree of the original vertices is more than two. Therefore, we give the different reduction for regular graphs, and hence we have:

Theorem 2. OneSDC on r-regular graphs is $\mathcal{NP}$-complete for $r \geq 3$.

Proof. It is obvious that OneSDC belongs to $\mathcal{NP}$. Then, we show that OneSDC is $\mathcal{NP}$-hard for r-regular graphs by giving a polynomial-time reduction from HC on 3-regular graphs which is known to be $\mathcal{NP}$-complete [12]. Our basic idea of the reduction is as follows: Let G be a 3-regular graph as an instance of HC, and let H be the r-regular graph which corresponds to G as the instance of OneSDC. That is, H is constructed so that G contains a Hamiltonian cycle if and only if H contains a single dominating cycle.

We first give the reduction for 3-regular graphs, and then modify it to one for general $r \geq 4$: Let G be a 3-regular graph, $V(G) = \{v_1, v_2, \ldots, v_n\}$ of n vertices, and $E(G) = \{e_1, e_2, \ldots, e_m\}$ of m edges. The corresponding graph H consists of (i) n subgraphs $V_1, V_2, \ldots, V_n$, called *vertex-gadgets*, which are associated with n vertices $v_1, v_2, \ldots, v_n$ in $V(G)$, respectively; and (ii) m subgraphs $E_1, E_2, \ldots, E_m$, called *edge-gadgets*, which are associated with m edges $e_1, e_2, \ldots, e_m$ in $E(G)$, respectively.

Below we construct each gadget and the corresponding H. See Fig. 1, (a) for G and (b) for H.

(i) For each i, $1 \leq i \leq n$, the i-th vertex-gadget V_i (middle gray disk in Fig. 1 (b)) contains three black vertices labeled by $v_{i,a}$, $v_{i,b}$, and $v_{i,c}$, and six white vertices.
(ii) For each j, $1 \leq j \leq m$, the j-th edge-gadget E_j contains only one edge $\{v_{a,b}, v_{b,a}\}$ between two vertex-gadgets V_a and V_b if a pair of vertices v_a and v_b are connected by the edge $e_j = \{v_a, v_b\}$ in G.

This completes the construction of the corresponding graph H. Clearly, this reduction can be done in polynomial time. Furthermore, H is 3-regular as shown in Fig. 1.

For a while, we make an observation on a subpath of the single dominating cycle going through the i-th vertex-gadget V_i. Let $C_H[V_i]$ be a path in a solution C_H (i.e., single dominating cycle) of OneSDC in V_i for $1 \leq i \leq n$. If any vertex in V_i is not included in the solution C_H, then only three vertices $v_{i,a}$, $v_{i,b}$, and $v_{i,c}$ in V_i can be dominated by C_H from the outside of V_i, which is a contradiction. Thus, $C_H[V_i] \neq \emptyset$ holds for every V_i and furthermore, $C_H[V_i]$ must dominate all the vertices in V_i. Since $C_H[V_i]$ connects to exactly two of the three edge-gadgets $\{v_{i,a}, v_{a,i}\}$, $\{v_{i,b}, v_{b,i}\}$, and $\{v_{i,c}, v_{c,i}\}$, the "dominating" path $C_H[V_i]$ in V_i should be classified into one of the following three sets of paths: $P_{x,y}$ from vertex x

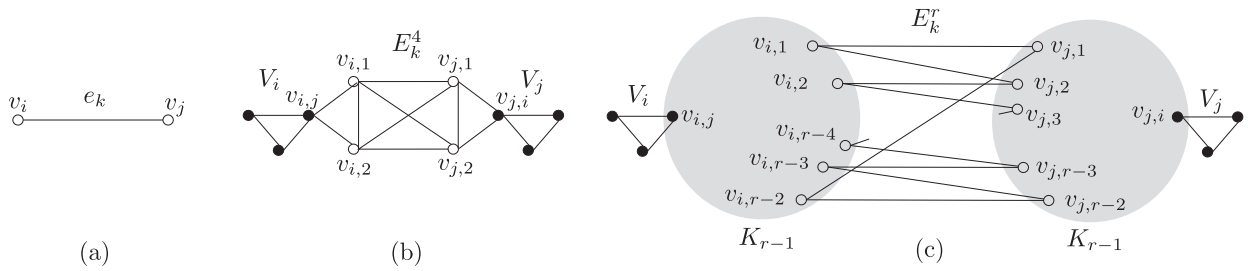


Fig. 2 (a) Graph G, (b) 4-regular graphs H_4, and (c) r-regular graph H_r ($r \geq 5$).

to vertex y where $(x, y) = \{(v_{i,a}, v_{i,b}), (v_{i,a}, v_{i,c}), (v_{i,b}, v_{i,c})\}$. Namely, for example, $P_{v_{i,a},v_{i,b}}$ is a set of dominating paths in V_i, $\langle v_{i,a}, u_1, u_2, u_4, u_6, v_{i,b}\rangle$, $\langle v_{i,a}, u_1, v_{i,c}, u_6, u_4, u_3, u_5, v_{i,b}\rangle$, $\langle v_{i,a}, u_1, u_2, u_4, u_3, u_5, v_{i,b}\rangle$, and so on.

Now we show that the graph G of HC contains a Hamiltonian cycle C_G if and only if the corresponding graph H of OneSDC contains a dominating cycle C_H. Basically, if an edge $\{v_a, v_i\}$ is included in C_G, then we choose the edge $\{v_{a,i}, v_{i,a}\}$ in C_H, and vice versa. Suppose that G contains a Hamiltonian cycle C_G. Then, we simply choose all edges in the edge-gadgets in H corresponding the edges in $E(C_G)$, and hence all vertex-gadgets are connected by chosen exactly two edge-gadgets in C_H. If the two edge-gadgets are $\{v_{a,i}, v_{i,a}\}$ and $\{v_{b,i}, v_{i,b}\}$ ($\{\{v_{a,i}, v_{i,a}\}, \{v_{c,i}, v_{i,c}\}\}$ and $\{\{v_{b,i}, v_{i,b}\}, \{v_{c,i}, v_{i,c}\}\}$, resp.), then a path in $P_{v_{i,a},v_{i,b}}$ ($P_{v_{i,a},v_{i,c}}$ and $P_{v_{i,b},v_{i,c}}$, resp.) which dominates all the vertices in V_i are chosen as $C_H[V_i]$. One can see that we can obtain a dominating cycle C_H in H by concatenating all $C_H[V_i]$'s for $1 \leq i \leq n$ and all edges in the edge-gadgets corresponding the edges in $E(C_G)$.

Suppose that H contains a dominating cycle C_H. As observed above, $C_H[V_i] \neq \emptyset$ holds for every V_i and $C_H[V_i]$ connects to exactly two of the three edge-gadgets $\{v_{i,a}, v_{a,i}\}$, $\{v_{i,b}, v_{b,i}\}$, and $\{v_{i,c}, v_{c,i}\}$. If $C_H[V_i]$ connects to $\{v_{i,a}, v_{a,i}\}$ and $\{v_{i,b}, v_{b,i}\}$ ($\{\{v_{i,a}, v_{a,i}\}, \{v_{i,c}, v_{c,i}\}\}$ and $\{\{v_{i,b}, v_{b,i}\}, \{v_{i,c}, v_{c,i}\}\}$, resp.), then two edges $\{v_a, v_i\}$ and $\{v_b, v_i\}$ ($\{\{v_a, v_i\}, \{v_c, v_i\}\}$ and $\{\{v_b, v_i\}, \{v_c, v_i\}\}$, resp.) incident to v_i are selected from the graph G of HC. Then, by concatenating such two edges incident to every vertex v_i for $1 \leq i \leq n$, we can construct a Hamiltonian cycle in G. This completes the proof for 3-regular graphs.

In the following, we show that the reduction for 3-regular graphs can be modified for r-regular graphs ($r \geq 4$).

(a) For each i, $1 \leq i \leq n$, if $r \geq 4$, the vertex-gadget V_i forms a triangle of three black vertices, as illustrated in Fig. 2 (b).

(b) For each edge $e_k = \{v_i, v_j\}$, $1 \leq k \leq m$,

- For $r = 4$, the k-th edge-gadget E_k^4 consists of four vertices $v_{i,1}$, $v_{i,2}$, $v_{j,1}$, and $v_{j,2}$, as illustrated in Fig. 2 (b). The graph $G[\{v_{i,1}, v_{i,2}, v_{j,1}, v_{j,2}\}]$ induced by those four vertices is a complete graph, K_4. Furthermore, $v_{i,1}$ and $v_{i,2}$ connect to $v_{i,j}$ in the i-th vertex-gadget V_i, and also $v_{j,1}$ and $v_{j,2}$ connect to $v_{j,i}$ in the j-th vertex-gadget V_j.
- For $r \geq 5$, the k-th edge-gadget E_k^r is a clique $K_{2(r-1)}$ of $2 \times (r-2)$ vertices, $\{v_{i,1}, \cdots, v_{i,r-2}\}$ and $\{v_{j,1}, \cdots, v_{j,r-2}\}$, as shown in Fig. 2 (c). $\{v_{i,1}, \cdots, v_{i,r-2}\}$ and $\{v_{j,1}, \cdots, v_{j,r-2}\}$ are connected to $v_{i,j}$ in the i-th vertex-gadget V_i and $v_{j,i}$ in the j-th vertex-gadget V_j, respectively. That is, the graphs induced by $\{v_{i,1}, \cdots, v_{i,r-2}\} \cup \{v_{i,j}\}$ and $\{v_{j,1}, \cdots, v_{j,r-2}\} \cup \{v_{j,i}\}$ are both cliques, K_{r-1}.

For $r = 4$, if edge e_k is included in a Hamiltonian cycle C_G in G, then we choose a path from $v_{i,j}$ to $v_{j,i}$ (or from $v_{j,i}$ to $v_{i,j}$), and the path dominates any other vertices in E_k^4. If edge e_k is not included in any Hamiltonian cycle in G, then vertices in E_k^4 cannot be dominated. The arguments for $r \geq 5$ are almost the same as above. □

Theorem 2 implies the following inapproximability:

Corollary 2. Let $\rho(n) \geq 1$ be any polynomial-time computable function. For r-regular graphs ($r \geq 3$), MinSDC admits no polynomial-time approximation algorithm with a factor of $\rho(n)$ unless $\mathcal{P} = \mathcal{NP}$. (The proof is very similar to one of Corollary 1 and thus omitted.)

4. Split Graphs

In this section, we consider the complexity of MinSDC on split graphs. Recall that a class of split graphs is a well-known subclass of chordal graphs. As mentioned before, MinSDC on split graphs is known to be $\mathcal{NP}$-hard [6], and moreover, MinSDC is $\mathcal{NP}$-hard even to approximate for chordal graphs as shown in the previous section. Fortunately, however, we first show that OneSDC on split graphs is tractable in the following.

Tractability of OneSDC. A split graph G is a graph for which the vertex set can be split into two portions, one including a clique set $Q \subseteq V(G)$, the other an independent set $I \subseteq V(G)$, i.e., $Q \cap I = \emptyset$ and $Q \cup I = V(G)$. It is known [15] that there is a linear-algorithm which can determine whether a given graph is split or not. Without loss of generality, assume that the clique graph $G[Q]$ has at least three vertices, $v_0, v_1, \cdots, v_{|Q|-1}$. Then, an (arbitrary) ordered sequence $\langle v_0, v_1, \cdots, v_{|Q|-1}, v_0\rangle$ of $|Q|$ vertices forms a cycle, say, C, i.e., C must be a Hamiltonian cycle in $G[Q]$. Since the Hamiltonian cycle C in the clique graph $G[Q]$ can dominate all vertices in the independent set I, it can be regarded as a single dominating cycle in G. Therefore, we can obtain

the following theorem:

Theorem 3. There exists a linear-time algorithm to solve OneSDC on split graphs.

Approximability of MinSDC. We provide a polynomial-time $(\ln n+1)$-approximation algorithm for MinSDC on split graphs with n vertices. The following lemma is quite trivial, but plays an important role to design the approximation algorithm:

Lemma 3. Suppose that a split graph G is split into two portions, the clique set Q and the independent set I. Then, if G has a single dominating cycle C of length ℓ such that $V(C) \cap I \neq \emptyset$, then we can find a shorter dominating cycle of length at most $\ell - 1$ in the split graph G.

Proof. Suppose that a dominating cycle of length ℓ, say, $C = \langle v_0, v_1, \cdots, v_{i-1}, v_i, v_{i+1}, \cdots, v_{\ell-1}, v_0\rangle$, has at least one vertex v_i in I. Then, for the subsequence $\langle v_{i-1}, v_i, v_{i+1}\rangle$ of C, two vertices v_{i-1} and v_{i+1} must be in Q since v_i is the vertex in the independent set I. Furthermore, there exists an edge $\{v_{i-1}, v_{i+1}\}$ since v_{i-1} and v_{i+1} are in the clique graph $G[Q]$. Since both v_{i-1} and v_{i+1} dominates v_i and also $V(C) \setminus \{v_i\}$ must dominate $V(G) \setminus (V(C) \setminus \{v_i\})$, the cycle $C' = \langle v_0, v_1, \cdots, v_{i-1}, v_{i+1}, \cdots, v_0\rangle$ of length $\ell - 1$ must be a shorter dominating cycle. □

By using Lemma 3 repeatedly, we can show that every vertex in an optimal dominating cycle is in the clique graph of the given split graph. If so, we can regard MinSDC on split graphs as the Minimum Unweighted Set Cover problem (MinSC for short) [13]: Given a set of n elements, $U = \{u_1, u_2, \cdots, u_n\}$, and a collection of m subsets of U, $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$, the goal of MinSC is to find a minimum cardinality collection $\mathcal{S}'$ of subsets from $\mathcal{S}$ such that $\mathcal{S}'$ covers all elements in U. A vertex $v \in Q$ and a vertex $u \in I$ in MinSDC correspond to a set S_v and an element e_u in MinSC, respectively. If there is an edge $\{u, v\}$, then $e_u \in S_v$. Then, if $\mathcal{S}' = \{S_{v_0}, S_{v_1}, \ldots, S_{v_0}\}$ is a set cover, then the corresponding cycle $C = \langle v_0, v_1, \cdots, v_0\rangle$ is a dominating cycle, and vice versa.

Theorem 4. There is a polynomial-time $(\ln n + 1)$-approximation algorithm for MinSDC on split graphs with n vertices.

Proof. Suppose that a split graph G is split into two portions, the clique set Q and the independent set I. At each step, the approximation algorithm greedily chooses a vertex, say, v of maximum degree in Q, then deletes v and its neighbor vertices $N(v) \cap I$, and repeats this process until all the vertices in I are deleted. It can be shown [5], [16], [19], [25] that the approximation ratio of the greedy algorithm is $(\ln n + 1)$. □

Inapproximability of MinSDC. Note that the $\mathcal{NP}$-hardness of MinSDC is previously proved by the polynomial-time reduction from the Minimum Vertex Cover problem [6]. In this section, for every $\varepsilon > 0$, we show the $(1 - \varepsilon)\ln n$-approximation hardness by the approximation-gap preserving reduction from the MinSC. Note that, very recently, Dinur and Steurer [9] show that it is $\mathcal{NP}$-hard to approximate MinSC to within a factor of $(1 - \varepsilon)\ln n$, by relaxing the previous assumption in [10].

Theorem 5. For split graphs, MinSDC admits no $(1-\varepsilon)\ln n$-approximation algorithm for every $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$.

Proof. We give a gap-preserving reduction from MinSC to MinSDC on split graphs. Let $U = \{u_1, u_2, \cdots, u_n\}$ and $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$ be an instance of MinSC. Without loss of generality, assume that $n \geq m$. The corresponding graph H of n vertices $\{u_1, u_2, \ldots, u_n\}$ in the independent set I, which are associated with n elements of U, respectively, and m vertices $\{s_1, s_2, \ldots, s_m\}$ in the clique graph, which are associated with m subsets $\{S_1, S_2, \cdots, S_m\}$ of set $\mathcal{S}$, respectively. If S_j has an element u_i in the instance MinSC, then s_j is connected to u_i in H. This completes the reduction.

Let OPT_{SC} (and OPT_{SDC}, resp.) denote the number of the subsets of an optimal solution of MinSC (and the length of the dominating cycle of an optimal solution of MinSDC, resp.). Let $g(n)$ be a parameter function of the instance of MinSC. Then, we can show that the above reduction satisfies the following: (1) if $OPT_{SC} \leq g(n)$, then $OPT_{SDC} \leq g(n)$, and (2) if $OPT_{SC} > g(n) \times (1 - \varepsilon)\ln n$, then $OPT_{SDC} > g(n)\times(1-\varepsilon)\ln n$ for a positive constant ε. Since the number of vertices in H is at most $2n$, the approximation gap is still $(1 - \varepsilon_1)\ln n$ for a small positive ε_1. □

5. Bounded Treewidth

In this section we show that there exists a linear-time algorithm for MinSDC if the input is restricted to a class of graphs with *bounded treewidth*, which includes, for example, classes of series-parallel, outerplanar, and p-outerplanar (for a fixed constant p) graphs.

Let $G = (V, E)$ be an n-vertex graph. A *tree decomposition* of a graph $G = (V, E)$ is a tree T in which each node $i \in T$ has an assigned set of vertices $X_i \subseteq V$, called a *bag*, such that $\bigcup_{i\in T} X_i = V$ with the following properties: (1) For all $\{u, v\} \in E$, there exists an $i \in T$ such that $\{u, v\} \subseteq X_i$; and (2) if $v \in X_i$ and $v \in X_j$, then $v \in X_k$ for all X_k on the path from X_i to X_j in T. The *width* of a tree decomposition is the size of the largest bag of T minus one, $\max_{i\in T} |X_i| - 1$. The *treewidth* k of a graph G is the minimum width over all possible tree decompositions of G. To distinguish between the vertices of the decomposition tree T and the vertices of the graph G, we refer to the vertices of T as *nodes*. Each node t of T corresponds to a subgraph $G_t = G[V_t]$ of G which is induced by the set V_t of vertices that are contained in the bag X_t and all bags of descendants of t in T. A *nice tree decomposition* is a rooted binary tree with five different types of nodes [1]: **(Root node)** the root r of T such that $X_r = \emptyset$; **(Leaf node)** a leaf ℓ of T with no children such that $X_\ell = \emptyset$; **(Introduce node)** a node t with exactly one child t' such that

$X_t = X_{t'} \cup \{v\}$ for some vertex $v \notin X_{t'}$. We say that the vertex v is introduced at t; **(Forget node)** a node t with exactly one child t' such that $X_t = X_{t'} \setminus \{w\}$ for some vertex $w \in X_{t'}$. We say that the vertex w is forgotten at t; **(Join node)** a node t with two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$.

Monadic second order logic on MinSDC. As a seminal result of Courcelle [7], it is known that every optimization problem that can be expressed in monadic second-order logic, so-called $\mathtt{MSO}_2$ formulas, can be solved for graphs with bounded treewidth in time linear in the number of vertices of the graph. Indeed, we can write a formula that is true in a graph G if and only if G admits a cycle C of length ℓ which can dominate all vertices in $V(G)$: First we consider a formula $\mathtt{inc}(v, e)$, which checks whether an edge e is incident with a vertex v. Then, we need to verify that (i) a subset C of edges induces a connected graph, (ii) every vertex in $V(C)$ is adjacent to exactly two different edges of C, and (iii) every vertex in $V(G)$ is in $V(C)$ or in $N(V(C))$. Consider the following three formulas:

$$\begin{aligned}
&\mathtt{ConnEdge}(C)\\
&= \forall_{S \subseteq V(C)} [(\exists_{u \in V(C)} u \in S \wedge \exists_{v \in V(C)} v \notin S)\\
&\qquad \Rightarrow (\exists_{e \in C} \exists_{u \in S} \exists_{v \notin S} \mathtt{inc}(u, e) \wedge \mathtt{inc}(v, e))]\\
&\mathtt{Deg2}(v, C)\\
&= \exists_{e_1, e_2 \in C} [(e_1 \neq e_2) \wedge \mathtt{inc}(v, e_1) \wedge \mathtt{inc}(v, e_2)\\
&\qquad \wedge (\forall_{e_3 \in C} \mathtt{inc}(v, e_3) \Rightarrow (e_1 = e_3 \vee e_2 = e_3))]\\
&\mathtt{Dominated}(v, C)\\
&= \exists_{u \in V(C)} [(u = v) \vee\\
&\qquad ((u \neq v) \wedge (\exists_{e \in E} \mathtt{inc}(u, e) \wedge \mathtt{inc}(v, e)))]
\end{aligned}$$

Here, $\mathtt{ConnEdge}(C)$ is a formula which checks whether the graph $(V(C), C)$ is connected, $\mathtt{Deg2}(v, C)$ verifies that a vertex v has exactly two adjacent edges belonging to C, and $\mathtt{Dominated}(v, C)$ verifies that a vertex v is in $V(C)$ or it has an edge $\{v, u\}$ for a vertex $u \in V(C)$. Therefore, the sentence "C is a single cycle of length ℓ which dominates all vertices in $V(G)$" is expressed by a formula, say, $\phi(C)$:

$$\begin{aligned}
\phi(C) = \exists_{e_1 \subseteq E(C)} \cdots \exists_{e_\ell \subseteq E(C)} &\mathtt{ConnEdge}(C)\\
&\wedge \forall_{v \in V(C)} \mathtt{Deg2}(v, C)\\
&\wedge \forall_{v \in V(G)} \mathtt{Dominated}(v, C)
\end{aligned}$$

From Courcelle's theorem, the shortest cycle C which dominates all vertices in graphs with bounded treewidth can be found in linear time. Unfortunately, however, the algorithm is very slow since the dependence on treewidth in the hidden constant factor of the running time is a tower of exponentials. From this reason, it is suggested [8] that Courcelle's theorem and its variants should be regarded primarily as classification tools and thus designing efficient dynamic-programming routines on tree decompositions requires "getting our hands dirty" and constructing the algorithm explicitly (see Sect. 7.4.2 in [8] for details). In the following, we design such a dynamic-programming algorithm explicitly.

Dynamic-programming algorithm. It is known that any graph of treewidth k has a nice tree-decomposition of width k [1]. Since a nice tree-decomposition of a graph G with bounded treewidth can be found in linear time [1], we may assume without loss of generality that both G and its nice tree-decomposition are given. Let G be a graph whose treewidth is bounded by a fixed constant k. Also, let a nice tree decomposition of G be represented by $(T, \{X_t\}_{t \in V(T)})$ or simply T.

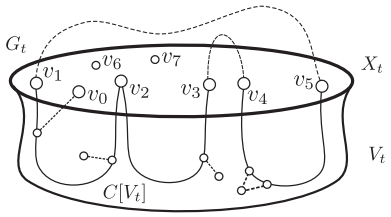


Fig. 3 Bag X_t for node t of T, and partial solution $C[V_t]$ in dominating cycle C.

Now consider an arbitrary dominating cycle C in G and the subgraph $C[V_t]$ in C, which is induced by the vertices in $V(C) \cap V_t$ for a node t of T. Then, $C[V_t]$ is a set of paths with endpoints in X_t. See Fig. 3 as an example. The vertex v_0 is dominated by C and thus $deg(C[V_t], v_0) = 0$. The vertices v_1 and v_3 are two endpoints of a subpath $P_{v_1,v_3} = \langle v_1, \cdots, v_2, \cdots, v_3 \rangle$, and v_2 is a middle vertex in P_{v_1,v_3}. The vertices v_4 and v_5 are two endpoints of another subpath $P_{v_4,v_5} = \langle v_4, \cdots, v_5 \rangle$. Thus, $deg(C[V_t], v_1) = deg(C[V_t], v_3) = deg(C[V_t], v_4) = deg(C[V_t], v_5) = 1$ and $deg(C[V_t], v_2) = 2$.

Let a labeling function f_t of a bag X_t be $f_t : X_t \to \{0, 1, 2, *\}$. Then, according to the labeling f_t, we partition a bag X_t into four subsets, $B_t^0 = \{v \mid f_t(v) = 0, v \in X_t\}$, $B_t^1 = \{v \mid f_t(v) = 1, v \in X_t\}$, $B_t^2 = \{v \mid f_t(v) = 2, v \in X_t\}$ and $B_t^* = \{v \mid f_t(v) = *, v \in X_t\}$:

- **Dominated** vertices, labeled 0. The meaning is that all *dominated* vertices are not contained in the partial solution (i.e., a part of the final dominating cycle) in G_t, and must be dominated by it.
- **Unsaturated** vertices, labeled 1. The meaning is that all *unsaturated* vertices have to be contained in the partial solution and their degrees in it must be exactly one.
- **Saturated** vertices, labeled 2. The meaning is that all *saturated* vertices have to be contained in the partial solution and their degrees in it must be two.
- **Undefined** vertices, labeled $*$. The meaning is that all *undefined* vertices are not contained in the partial solution, but currently do not have to be dominated by the partial solution, i.e., the vertices currently labeled $*$ would be eventually put into the solution, or dominated by the solution. For example, a (current) labeling function $f_t(v_6) = f_t(v_7) = *$ possibly implies the partial solution in G_t, illustrated in Fig. 3.

Let M_t be a matching of unsaturated vertices in X_t such that

$$\begin{aligned}
M_t = \{\{u, v\} \mid &\text{the partial solution includes a path}\\
&\text{from } u \text{ to } v \text{ for } u, v \in X_t\}.
\end{aligned}$$

Take a look at Fig. 3 again. Since the subgraph $C[V_t]$ in-

cludes a subpath P_{v_1,v_3}, the matching $\{v_1, v_3\}$ is in M_t.

For the labeling f_t, i.e., the partition $(B_t^0, B_t^1, B_t^2, B_t^*)$, and matching M_t of X_t, we denote by $c[t; (B_t^0, B_t^1, B_t^2, B_t^*), M_t]$, or simply $c[t; f_t, M_t]$, the minimum number of vertices in a subgraph $C[V_t]$ such that

- $C[X_t] = B_t^1 \cup B_t^2$, which is the set of saturated and unsaturated vertices of X_t, i.e., the set of vertices in the partial solution.
- Each vertex of $V_t \setminus B_t^*$ either is in $C[V_t]$ or is adjacent in G_t to a vertex of $C[V_t]$. That is, $C[V_t]$ dominates all vertices of $V_t \setminus V(C)$ in the subgraph G_t, except possibly some undefined vertices in X_t.

We say such a subgraph $C[V_t]$ a *compatible subgraph* for t, f_t and M_t. If no compatible subgraph for t, f_t and M_t exists, we let $c[t; f_t, M_t] = +\infty$. Our algorithm computes $c[t; f_t, M_t]$ for each node t of T and all pairs (f_t, M_t) for X_t, from leaves of T to the root r of T, by means of dynamic-programming. Then, since $G_r = G$ for the root r of T, one can compute the minimum dominating cycle in G as the value of $c[r; \emptyset, \emptyset] \equiv c[r; (\emptyset, \emptyset, \emptyset, \emptyset), M_t]$. This is because we have $X_r = \emptyset$, which means that for X_r we have only the empty function and the empty matching.

Now we explain how to compute $c[t; f_t, M_t]$ for each node t of T from the leaves to the root of T.

(1) Leaf node of T: For a leaf node t we have that $X_t = \emptyset$. Therefore, there is only one possibility, empty function and then empty matching. We have $c[t; \emptyset, \emptyset] = 0$.

(2) Introduce node of T: Let t be an introduce node with a child t' such that $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$. Note that since v is introduced by X_t, every edge in G_t incident to v is contained in $G[X_t]$, that is, $N(G_t, v) \subseteq X_t$. We have to consider the following labelings on v:

(i) $f_t(v) = 0$, i.e., $B_t^0 = B_{t'}^0 \cup \{v\}$, and the labeling f_t is the same as $f_{t'}$ for $X_{t'}$. Then, we set $c[t; f_t, M_t] = c[t'; f_{t'}, M_{t'}]$.

(ii) $f_t(v) = *$, i.e., $B_t^* = B_{t'}^* \cup \{v\}$, and the labeling f_t is the same as $f_{t'}$ for $X_{t'}$. Then, we set $c[t; f_t, M_t] = c[t'; f_{t'}, M_{t'}]$.

(iii) $f_t(v) = 1$, and the labeling f_t is almost the same as $f_{t'}$ for $X_{t'}$, but it is changed for a neighbor vertex u in $N(G(X_t), v)$:

- If v is connected to a vertex u in $B_{t'}^*$, then $B_t^1 = B_{t'}^1 \cup \{v, u\}$, $B_t^* = B_{t'}^* \setminus \{u\}$ and $M_t = M_{t'} \cup \{\{u, v\}\}$. We set $c[t; f_t, M_t] = c[t'; f_{t'}, M_{t'}] + 1$.
- If v is connected to a vertex u in $B_{t'}^1$, then $B_t^1 = (B_{t'}^1 \setminus \{u\}) \cup \{v\}$, $B_t^2 = B_{t'}^2 \cup \{u\}$ and $M_t = (M_{t'} \setminus \{\{u, u'\}\}) \cup \{\{v, u'\}\}$, assuming the partial solution at t' includes a path from u to u'. We set $c[t; f_t, M_t] = c[t'; f_{t'}, M_{t'}] + 1$.

(iv) $f_t(v) = 2$, and the labeling f_t is almost the same as $f_{t'}$ for $X_{t'}$, but it is changed for two neighbor vertices u and w in $N(G(X_t), v)$:

- If v is connected to two vertices u and w in $B_{t'}^*$, then $B_t^2 = B_{t'}^2 \cup \{v\}$, $B_t^* = B_{t'}^* \setminus \{u, w\}$, $B_t^1 = B_{t'}^1 \cup \{u, w\}$, and $M_t = M_{t'} \cup \{\{u, w\}\}$. We set $c[t; f_t, M_t] = c[t'; f_{t'}, M_{t'}] + 3$.
- If v is connected to a vertex u in $B_{t'}^*$ and a vertex w in $B_{t'}^1$, then $B_t^2 = B_{t'}^2 \cup \{v, w\}$, $B_t^* = B_{t'}^* \setminus \{u\}$, $B_t^1 = (B_{t'}^1 \setminus \{w\}) \cup \{u\}$, and $M_t = (M_{t'} \setminus \{\{w, w'\}\}) \cup \{\{u, w'\}\}$, assuming the partial solution at t' includes a path from w to w'. We set $c[t; f_t, M_t] = c[t'; f_{t'}, M_{t'}] + 2$.
- If v is connected to two vertices u and w in $B_{t'}^1$ from different paths $P_{u,u'}$ and $P_{w,w'}$, then $B_t^2 = B_{t'}^2 \cup \{v, u, w\}$, $B_t^1 = B_{t'}^1 \setminus \{u, w\}$, and $M_t = (M_{t'} \setminus \{\{u, u'\}, \{w, w'\}\}) \cup \{\{u', w'\}\}$. We set $c[t; f_t, M_t] = c[t'; f_{t'}, M_{t'}] + 1$.
- If v is connected to two vertices u and w in $B_{t'}^1$ from the same path $P_{u,w}$, then $B_t^2 = B_{t'}^2 \cup \{v, u, w\}$, $B_t^1 = B_{t'}^1 \setminus \{u, w\}$, and $M_t = M_{t'} \setminus \{\{u, w\}\}$. We set $c[t; f_t, M_t] = c[t'; f_{t'}, M_{t'}] + 1$. Note that in this case we get a single dominating cycle for G_t (and for G).

(3) Forget node of T: Let t be a forget node with a child t' such that $X_t = X_{t'} \setminus \{v\}$ for some $v \in X_{t'}$. Let $f_{t'}^0(v)$ be a labeling which is almost the same as $f_t(v)$ but v is labeled by 0. Also, $f_{t'}^2(v)$ be a labeling which is almost the same as $f_t(v)$ but v is labeled by 2. Note that the definition of compatible subgraphs for t, f_t, and M_t requires that the forgotten vertex v is dominated or saturated. That is, every compatible subgraph $C[V_t]$ for t, f_t and M_t is also compatible for t', $f_{t'}^0$, and $M_{t'}$ (in the case where v is in $C[V_t]$) or t', $f_{t'}^2$, and $M_{t'}$ (in the case where v is not in $C[V_t]$), and vice versa. Therefore, we set

$$c[t; f_t, M_t] = \min\left\{c[t'; f_{t'}^0, M_{t'}], c[t'; f_{t'}^2, M_{t'}]\right\}.$$

(4) Join node of T: Let t be a join node with children t_1 and t_2. Note that $X_t = X_{t_1} = X_{t_2}$ and there is no edge joining a vertex in $G_{t_1} \setminus X_{t_1}$ and one in $G_{t_2} \setminus X_{t_2}$. Let f_{t_1} and M_{t_1} be a labeling and a matching for X_{t_1}. Also, let f_{t_2} and M_{t_2} be a labeling and a matching for X_{t_2}. Roughly speaking, in the computation for t, we merge two partial solutions, a set of subpaths going through in G_{t_1} and the other set of subpaths going through in G_{t_1}. We have to consider the following labelings and matchings:

(i) We set $f_t(v) = 2$ if a pair of two labelings $(f_{t_1}(v), f_{t_2}(v))$ is as follows: $(f_{t_1}(v), f_{t_2}(v)) \in \{(2, 0), (2, *), (0, 2), (*, 2)\}$.

(ii) We set $f_t(v) = 2$, $f_t(u) = 1$, $f_t(w) = 1$ and $\{u, w\} \in M_t$ for $v, u, w \in X_t$ if $\{v, u\} \in M_{t_1}$ and $\{v, w\} \in M_{t_2}$, i.e., $f_{t_1}(v) = f_{t_1}(u) = f_{t_2}(v) = f_{t_2}(w) = 1$. That is, the partial solution includes a path $P_{u,w}$ from u to w.

(iii) We set $f_t(v) = *$ if a pair of two labelings $(f_{t_1}(v), f_{t_2}(v)) = (*, *)$.

(iv) We set $f_t(v) = 0$ if a pair of two labelings $(f_{t_1}(v), f_{t_2}(v)) \in \{(0, *), (*, 0)\}$.

Then, we set

$$\begin{aligned} &c[t; f_t, M_t] \\ &= \min_{f_{t_1}, f_{t_2}} \left\{c[t_1; f_{t_1}, M_{t_1} + c[t_2; f_{t_2}, M_{t_2}] - \mid B_1^{t_1} \cap B_2^{t_2} \mid\right\}. \end{aligned}$$

Running time. Recall that a given graph G is of treewidth bounded by a fixed constant k, and hence each bag X_t of T contains at most $k + 1$ vertices. The labeling function f_t has four possibilities $\{0, 1, 2, *\}$, i.e., the number of possible labelings on f_t is at most 4^{k+1}. An upper bound for the number

of possible matchings on M_t is at most $(k+2)! = O(k^k \cdot e^{\sqrt{k}})$. (1) We spend a constant time at every leaf node of T. The computation (2) for every introduced node of T requires $O(k^2 \cdot 4^k \cdot k^k)$ time, and (3) for every forget node of T, $O(4^k \cdot k^k)$ time. (4) We spend the maximum time for the computation on every joint node of T, which is at most $O(k^2 \cdot (4^k \cdot k^k)^2) = O(1)$ time for a fixed constant k. Thus, the above algorithm runs in linear time for graphs with bounded treewidth:

Theorem 6. There exists an algorithm to solve MinSDC on graphs of treewidth k in $O(k^2 \cdot (4^k \cdot k^k)^2 \cdot n)$ time.

Acknowledgments

This work is partially supported by JSPS KAKENHI JP15J05484, JP26330017 and JP17K00016. The authors would like to thank the anonymous reviewers for their detailed comments and suggestions that helped to improve the presentation of the paper.

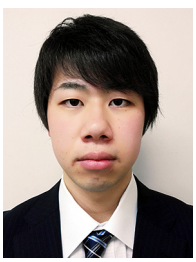
References

[1] N. Betzler, R. Niedermeier, and J. Uhlmann, "Tree decompositions of graphs: Saving memory in dynamic programming," Discrete Optimization, vol.3, no.3, pp.220–229, 2006.
[2] A. Brandstät, V.B. Le, and J.P. Spinrad, Graph Classes: A Survey, SIAM, 1987.
[3] W.-P. Chin and S. Ntafos, "Optimum watchman routes," Inform. Proc. Lett., vol.28, no.1, pp.39–44, 1988.
[4] V. Chvátal, "A combinatorial theorem in plane geometry," J. Combinatorial Theory, B, vol.18, no.1, pp.39–41, 1975.
[5] V. Chvátal, "A greedy heuristic for the set-covering problem," Math. Oper. Res., vol.4, no.3, pp.233–235, 1979.
[6] C.J. Colbourn and L.K. Stewart, "Dominating cycles in series-parallel graphs," Ars Combin., vol.19A, pp.107–112, 1985.
[7] B. Courcelle, "The monadic second-order logic of graphs. I. Recognizable sets of finite graphs," Inf. Comput., vol.85, no.1, pp.12–75, 1990.
[8] M. Cygan, F.V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, Parameterized Algorithms, Springer, 2015.
[9] I. Dinur and D. Steurer, "Analytical approach to parallel repetition," STOC '14, pp.624–633, 2014.
[10] U. Feige, "A threshold of $\ln n$ for approximating set cover," J. ACM, vol.45, no.4, pp.634–652, 1998.
[11] M.R. Garey, D.S. Johnson, and L. Stockmeyer, "Some simplified NP-complete problems," STOC '74, pp.47–63, 1974.
[12] M.R. Garey, D.S. Johnson, R.E. Tarjan, "The planar Hamiltonian circuit problem is NP-complete," SIAM J. Comput, vol.5, no.4, pp.704–714, 1976.
[13] M.R. Garey and D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, Freeman and Company, 1979.
[14] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.
[15] P.L. Hammer, B. Simeonem "The splittance of a graph," Combinatorica, vol.1, no.3, pp.275–284 1981.
[16] D.S. Johnson, "Approximation algorithms for combinatorial problems," J. Comput. System Sci., vol.9, no.3, pp.256–278, 1974.
[17] J.M. Keil and D. Schaefer, "An optimal algorithm for finding dominating cycles in circular-arc graph," Discrete Applied Mathematics, vol.36, no.1, pp.25–34, 1992.
[18] M.S. Krisnamoorthy, "An NP-hard problem in bipartite graphs," SIGACT News, vol.7, no.1, pp.26–26, 1976.
[19] L. Lovász, "On the ratio of optimal integral and fractional covers," Discrete Math., vol.13, no.4, pp.383–390, 1975.
[20] H. Müller, "Hamiltonian circuits in chordal bipartite graphs," Discrete Math., vol.156, no.1-3, pp.291–298, 1996.
[21] C.St.J.A. Nash-Williams, "Hamiltonian circuits in graphs and digraphs," The Many Facets of Graph Theory, Lecture Notes in Mathematics, pp.237–243, Springer, Berlin, Heidelberg, 1969.
[22] J. O'Rourke, Art gallery theorems and algorithms, Oxford University Press, 1987.
[23] A. Proskurowski, "Minimum dominating cycles in 2-trees," Int. J. Comp. & Info. Sci., vol.8, no.5, pp.405–417, 1979.
[24] A. Proskurowski and M.M. Sysło, "Minimum dominating cycles in outerplanar graphs," Int. J. Comp. & Info. Sci., vol.10, no.2, pp.127–139, 1981.
[25] P. Slavík, "A tight analysis of the greedy algorithm for set cover," J. Algorithms, vol.25, no.2, pp.237–254, 1997.

Hiroshi Eto is Assistant Professor of Faculty of Economics at Kyushu University. Received B.Eng., M.Eng., and Dr.Eng. degrees in 2011, 2013, and 1016, respectively. His research interest are in the area of algorithms and complexity. IPSJ, ORSJ.

Hiroyuki Kawahara received the B.Eng. and M.Eng. degrees from Kyushu Institute of Technology in 2015 and 2017, respectively.

Eiji Miyano received the B.Eng., M.Eng., and Dr.Eng. degrees in computer science from Kyushu University in 1991, 1993, 1995, respectively. He is currently a professor of the Department of Systems Design and Informatics, Kyushu Institute of Technology.



Natsuki Nonoue received the B.Eng. degree from Kyushu Institute of Technology in 2017. He is currently a graduate student of the Department of Systems Design and Informatics, Kyushu Institute of Technology.