Computational Study for Planar Connected Dominating Set Problem

Marjan Marzban[1], Qian-Ping Gu[1], and Xiaohua Jia[2]

[1] School of Computing Science, Simon Fraser University, Burnaby BC Canada
{mmarzba,qgu}@cs.sfu.ca
[2] Department of Computer Science, City University of Hong Kong
csjia@cityu.edu.hk

Abstract. The connected dominating set (CDS) problem is a well studied NP-hard problem with many important applications. Dorn et al. [ESA2005, LNCS3669,pp95-106] introduce a new technique to generate $2^{O(\sqrt{n})}$ time and fixed-parameter algorithms for a number of non-local hard problems, including the CDS problem in planar graphs. The practical performance of this algorithm is yet to be evaluated. We perform a computational study for such an evaluation. The results show that the size of instances can be solved by the algorithm mainly depends on the branchwidth of the instances, coinciding with the theoretical result. For graphs with small or moderate branchwidth, the CDS problem instances with size up to a few thousands edges can be solved in a practical time and memory space. This suggests that the branch-decomposition based algorithms can be practical for the planar CDS problem.

Keywords: Branch-decomposition based algorithms, CDS problem, planar graphs, fixed-parameter algorithms, computational study.

1 Introduction

In this paper, graphs are undirected, simple and finite unless otherwise stated. Let G be a graph with vertex set $V(G)$ and edge set $E(G)$. A dominating set D of G is a subset of $V(G)$ such that for every vertex $u \in V(G)$, $u \in D$ or u is incident to a vertex $v \in D$. The *dominating number* of G, denoted by $\gamma(G)$, is the minimum size of a dominating set of G. The dominating set problem is to decide if $\gamma(G) \leq k$ for a given G and integer k. The dominating set problem is a core NP-complete problem in combinatorial optimization [15]. The rich literature of algorithms and complexity of dominating set problem can be found in [20,19].

A subset D of $V(G)$ is a connected dominating set (CDS) of G if D is a dominating set of G and the subgraph $G[D]$ induced by D is connected. The *connected dominating number* of G, denoted by $\gamma_c(G)$, is the minimum size of a CDS of G. The CDS problem is to decide if $\gamma_c(G) \leq k$ for a given G and integer k. The optimization version of the CDS problem is to find a minimum CDS of an input graph. The CDS problem is an important variant of the dominating set problem and has wide practical applications in wireless ad hoc or sensor

W. Wu and O. Daescu (Eds.): COCOA 2010, Part II, LNCS 6509, pp. 107–116, 2010.

networks such as virtual backbone construction [6], energy efficient routing and broadcasting [5]. Notice that $\gamma(G) \leq \gamma_c(G) \leq 3\gamma(G) - 2$.

The CDS problem is NP-complete [15]. Approximation algorithms and exact fixed parameter algorithms have been main topics in the algorithmic research for the CDS problem. For arbitrary graphs, there are $2(1+\ln \Delta)$- and $(\ln \Delta + 3)$-approximation algorithms for the CDS problem, where Δ is the maximum vertex degree of the input graph [18]; the CDS problem is not approximable within a factor of $(1-\epsilon)\ln \Delta$ for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{\log \log n})$ [18]; and the CDS problem is fixed-parameter intractable unless the parameterized complexity classes collapse [11,12]. The CDS problem remains NP-complete if the input graphs are restricted to planar [15]. However, the planar CDS problem admits a PTAS [7], and is fixed parameter tractable [9,10].

Recently, significant progresses have been made on the fixed-parameter algorithms for the planar dominating set problem [13,8] and practical performance of those algorithms have been reported [21]. The notions of tree/branch-decompositions introduced by Robertson and Seymour [23,24,25] play a central role in those algorithms. Although the dominating set problem and the CDS problem are closely related, they have different properties from the tree/branch-decomposition based algorithm point of view. In particular, the techniques used to solve the dominating set problem do not seem to work for the CDS problem. One of the main reasons of such discrepancy is that *connectivity* is a *non-local property* (see Section 3 for more details).

Along the lines to clear the hurdles caused by the non-local property, Dorn et al. [9,10] propose a new technique to design sub-exponential time exact algorithms for the non-local problems in planar graphs. This new technique is based on the geometric properties of branch-decomposition of graphs with a planar embedding in a sphere and the properties of non-crossing partitions in the embedding. Based on this new technique, they show that many non-local problems in planar graphs can be solved in $2^{O(\sqrt{n})}$ time [9,10]. Especially, they give an algorithm (called DPBF Algorithm in what follows) which solves the planar CDS problem in $O(2^{O(\mathrm{bw}(G))}n + n^3)$ and $O(2^{9.822\sqrt{n}}n + n^3)$ time [10]. The constant in $O(\mathrm{bw}(G))$ is not explicitly given in [9,10]. By a more careful analysis, it can be shown that DPBF Algorithm solves the planar CDS problem in $O(2^{4.618\mathrm{bw}(G)}n + n^3)$ and $O(2^{9.8\sqrt{n}}n + n^3)$ time. The running time can be further improved to $O(2^{3.812\mathrm{bw}(G)}n + n^3)$ and $O(2^{8.088\sqrt{n}}n + n^3)$ if the fast distance matrix multiplication is applied [8]. It is known that $\mathrm{bw}(G) \leq 3\sqrt{4.5\gamma(G)}$ for planar graph G [14,13]. Since $\gamma(G) \leq \gamma_c(G)$, the planar CDS problem admits an $O(2^{24.257\sqrt{\gamma_c(G)}}n + n^3)$ time fixed-parameter algorithm.

Because of the applications of the planar CDS problem in wireless networks, practically efficient exact algorithms for the planar CDS problem are of great interests for those applications. DPBF Algorithm suggests theoretically an efficient exact approach for the CDS problem in planar graphs of small branchwidth. However, the practical performance of the algorithm is yet to be evaluated. In this paper, we perform a computational study to evaluate DPBF Algorithm.

We also apply the recent result on the kernelization for the planar CDS problem in our study. A linear size kernel of a graph G for the CDS problem is a subgraph H of G with $O(\gamma_c(G))$ vertices and $\gamma_c(H) \leq \gamma_c(G)$ such that a minimum CDS of G can be produced efficiently from a minimum CDS of H. It is known that the planar CDS problem admits a linear size kernel and such a kernel can be computed in $O(n^3)$ time [16]. Applying the algorithm of [16] to shrink the input graph G into a linear size kernel H, we get an $O(2^{24.257\sqrt{\gamma_c(G)}}\gamma_c(G) + n^3)$ time algorithm for the planar CDS problem.

The computational study is performed on several classes of planar graphs that cover a wide range of planar graphs. The results show that the conventional version of DPBF Algorithm is more efficient than the version of using fast distance matrix multiplication even though the latter has a better theoretical running time because the fast distance matrix multiplication itself is not practical. The size of instances that can be solved in a practical time and memory space mainly depends on the branchwidth of the kernels of the instances. This coincides with the theoretical running time of DPBF Algorithm.

The computational study gives a concrete example on using the branch-decomposition based algorithms for solving important non-local problems in planar graphs and shows that the planar CDS problem can be solved in practice for a wide range of graphs. This work provides a tool for computing the optimal CDS of planar graphs and may bring the sphere-cut decomposition and noncrossing partitions based approach closer to practice.

In the rest of the paper, Section 2 provides necessary definitions. We describe DPBF Algorithm and our implementation of it in Section 3. Computational results are reported in Section 4 and the final section concludes the paper.

2 Preliminaries

A graph $G(V, E)$ consists of a set $V(G)$ of vertices and a set $E(G)$ of edges. A graph H is a subgraph of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. For a subset $A \subseteq E(G)$ ($U \subseteq V(G)$), we denote by $G[A]$ ($G[U]$) the subgraph of G induced by A (U).

For a graph G and a subset $A \subseteq E(G)$ of edges, we denote $E(G) \setminus A$ by $\overline{A}$ when G is clear from the context. A separation of graph G is a pair $(A, \overline{A})$ of subsets of $E(G)$. For each $A \subseteq E(G)$, we denote by $\partial(A)$ the vertex set $V(A) \cap V(\overline{A})$. The *order* of separation $(A, \overline{A})$ is $|\partial(A)| = |\partial(\overline{A})|$.

A *branch-decomposition* of graph G [25] is a pair (ϕ, T) where T is a tree each internal node of which has degree 3 and ϕ is a bijection from the set of leaves of T to $E(G)$. Consider a link e of T and let L_1 and L_2 denote the sets of leaves of T in the two respective subtrees of T obtained by removing e. We say that the separation $(\phi(L_1), \phi(L_2))$ is induced by this link e of T. We define the width of the branch-decomposition (ϕ, T) to be the largest order of the separations induced by links of T. The *branchwidth* of G, denoted by $\mathrm{bw}(G)$, is the minimum width of all branch-decompositions of G. In the rest of this paper,

we identify a branch-decomposition (ϕ, T) with the tree T, leaving the bijection implicit and regarding each leaf of T as an edge of G.

Let Σ be a fixed sphere. A set P of points in Σ is a *topological segment* of Σ if it is homeomorphic to an open segment $\{(x, 0)|0 < x < 1\}$ in the plane. For a topological segment P, we denote by $\overline{P}$ the closure of P and $\mathrm{bd}(P) = \overline{P} \setminus P$ the two *end points* of P. A planar embedding of a graph G is a mapping $\rho : V(G) \cup E(G) \to \Sigma \cup 2^{\Sigma}$ satisfying the following properties.

- For $u \in V(G)$, $\rho(u)$ is a point of Σ, and for distinct $u, v \in V(G)$, $\rho(u) \neq \rho(v)$.
- For each edge $e = \{u, v\}$ of $E(G)$, $\rho(e)$ is a topological segment with two end points $\rho(u)$ and $\rho(v)$.
- For distinct $e_1, e_2 \in E(G)$, $\overline{\rho(e_1)} \cap \overline{\rho(e_2)} = \{\rho(u)|u \in e_1 \cap e_2\}$.

A graph is planar if it has a planar embedding. A plane graph is a pair (G, ρ), where ρ is a planar embedding of G. We may simply use G to denote the plane graph (G, ρ), leaving the embedding ρ implicit. We do not distinguish a vertex v (resp. an edge e) from its embedding $\rho(v)$ (resp. $\rho(e)$) when there is no confusion.

Let G be a plane graph. We say that a curve μ on the sphere Σ is *normal* if μ does not intersect with itself or any edge of G. A *noose* of G is a closed normal curve on Σ. Let ν be a noose of G and let R_1 and R_2 be the two open regions of the sphere separated by ν. Then, ν induces a separation $(A, \overline{A})$ of G, with $A = \{e \in E(G) \mid \rho(e) \subseteq R_1\}$ and $\overline{A} = \{e \in E(G) \mid \rho(e) \subseteq R_2\}$. We also say that noose ν induces edge-subset A of G if ν induces a separation $(A, \overline{A})$ having A on one side. We call a separation or an edge-subset *noose-induced* if it is induced by some noose. A branch-decomposition T of G is a *sphere-cut decomposition* if every separation induced by a link of T is noose-induced [9,10]. It is known that every plane graph G has an optimal branch-decomposition (of width $\mathrm{bw}(G)$) that is a sphere-cut decomposition and such a decomposition can be found in $O(n^3)$ time [26,17].

3 Algorithm for Planar CDS Problem

DPBF Algorithm uses the branch-decomposition based approach which has two major steps: (1) compute a branch-decomposition T of the input graph, and (2) apply dynamic programming method based on T to solve the problem. A link e of T is called a *leaf link* if e contains a leaf node of T, otherwise called an *internal link*. To solve an optimization problem P in Step (2), T is first converted to a rooted binary tree by replacing a link $\{x, y\}$ of T with three links $\{x, z\}, \{y, z\}, \{z, r\}$, where z and r are new nodes to T, r is the root, and $\{z, r\}$ is an internal link. A link e' (resp. a node x) is called a *descendant link* (resp. *descendant node*) of link e if e is in the path from e' (resp. x) to the root r of T. For a link e of T, let $(A_e, \overline{A_e})$ be the separation induced by e with A_e the set of leaf nodes of T (set of edges of G) that are descendant nodes of e. For a leaf link e, all possible partial solutions of P in the subgraph $G[A_e]$ can be computed by enumeration. For an internal link e of T, e has two child links e_1 and e_2. Notice that $A_e = A_{e_1} \cup A_{e_2}$. All possible partial solutions in the subgraph $G[A_e]$ are computed by merging the partial solutions in $G[A_{e_1}]$ and those in $G[A_{e_2}]$.

A problem P is known having a *local structure*, if a partial solution of P in $G[A_e]$ can be identified by a fixed number of states of each vertex in $\partial(A_e)$, and all partial solutions in $G[A_e]$ can be computed from the states of the vertices in $\partial(A_{e_1})$ and those of the vertices in $\partial(A_{e_2})$. The local structure is a key condition for the branch-decomposition based algorithm to solve P in $O(2^{O(\mathrm{bw}(G))}n^{O(1)})$ time. However for the CDS problem, the connectivity information in a partial solution in $G[A_e]$ may not be expressed by a fixed number of states of each vertex of $\partial(A_e)$. In the merge step, the structures of the partial solutions in the entire subgraphs $G[A_{e_1}]$ and $G[A_{e_2}]$ may have to be checked. Because of this, the CDS problem is known having a *non-local structure*.

Dorn et al. give a new technique which makes the branch-decomposition based approach applicable to many problems with the non-local structure in planar graphs [9,10]. This new technique is based on two observations. One is the geometric property of the sphere-cut decomposition T of plane graph G: For any link e of T and the separation $(A_e, \overline{A_e})$ induced by e, there is a noose ν_e such that ν_e induces $(A_e, \overline{A_e})$, ν_e partitions the sphere Σ into two regions, all edges of A_e are in one region, and all edges of $\overline{A_e}$ are in the other region. Notice that ν_e intersects all vertices of $\partial(A_e)$. The other observation is known as the non-crossing partitions: Let $P_1, ..., P_r$ be the subsets of A_e such that $G[P_i]$ is connected for each $1 \leq i \leq r$ and $G[P_i \cup P_j]$ is not connected for every pair of $1 \leq i \neq j \leq r$. We call $P_1, ..., P_r$ *disjoint components*. Two components P_i and P_j are called *crossing* if there are $u, u' \in V(P_i) \cap \partial(A_e)$ and $v, v' \in V(P_j) \cap \partial(A_e)$ such that the four vertices appear on ν_e in the orders u, v, u', v', otherwise *non-crossing*. Notice that if P_i and P_j are crossing then $G[P_i \cup P_j]$ is connected because $G[A_e]$ is a plane graph. So, any pair of disjoint components are non-crossing. The sphere-cut decomposition and the non-crossing partitions make it possible to compute the partial solutions in $G[A_e]$ by only looking at the local structures of partial solutions in $G[A_{e_1}]$ at $\partial(A_{e_1})$ and those in $G[A_{e_2}]$ at $\partial(A_{e_2})$.

For a minimum CDS D of G, the subgraph $G[D \cap V(A_e)]$ of $G[A_e]$ induced by D consists of disjoint components $P_1, ..., P_r$ with $|V(P_i) \cap \partial(A_e)| \geq 1$ for every $1 \leq i \leq r$. We assume the vertices of $\partial(A_e)$ are indexed as $u_1, u_2, ..., u_k$ in the clockwise order as they appear in the noose ν_e. If $|V(P_i) \cap \partial(A_e)| \geq 2$, we call the vertex of $V(P_i) \cap \partial(A_e)$ with the smallest index the *small end*, the vertex of $V(P_i) \cap \partial(A_e)$ with the largest index the *large end* and other vertices of $V(P_i) \cap \partial(A_e)$ the *middle vertices* of P_i. In DPBF Algorithm, each vertex $u \in \partial(A_e)$ is given one of the following six colors.

- Color 0, u does not appear in any P_i and is dominated by some vertex of $D \cap V(A_e)$.
- Color $\hat{0}$, u does not appear in any P_i and is not dominated by any vertex of $D \cap V(A_e)$.
- Color $1_{[}$, u is the small end of some P_i.
- Color $1_{]}$, u is the large end of some P_i.
- Color 1^*, u is a middle vertex of some P_i.
- Color $\hat{1}$, u is the only vertex of some $V(P_i) \cap \partial(A_e)$.

From the geometric property of sphere-cut decomposition and the non-crossing partitions, each partial solution $P_1, ..., P_r$ can be identified by a coloring of $\{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^{|\partial(A_e)|}$.

We implemented DPBF Algorithm together with a pre-processing step which reduces the input graph to a linear size kernel. There are three major steps in our implementation. Let G be a plane graph of n vertices.

Step I: Compute a kernel H of G with $|V(H)| = O(\gamma_c(G))$. This can be done in $O(n^3)$ time [16].

Step II: Compute a sphere-cut decomposition T of H with width $\mathrm{bw}(H)$. This can be done in $O((\gamma_c(H))^3)$ time [26,17].

Step III: Compute a minimum CDS D of H using the dynamic programming method based on T and compute a minimum CDS of G from D.

We use 1 to express the numerical value of $1_], 1_[, 1^*, \hat{1}$. Let $b = |\partial(A_e)|$. We call $0, \hat{0}, 1$ *basic colors* and a $\lambda \in \{0, \hat{0}, 1\}^b$ a *basic-coloring* of $\partial(A_e)$.

For a coloring $\eta \in \{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^b$, we denote by $D_e(\eta)$ the partial solution identified by η with the minimum number of black vertices. In the merge step for the link $e = \{z, r\}$ incident to the root node r, we check the connectivity of $H[D_e(\eta)]$. A $D_e(\eta)$ with the minimum cardinality and $H[D_e(\eta)]$ connected is a minimum CDS of H. For $\eta \in \{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^b$, we define $a_e(\eta) = |D_e(\eta)|$ if η identifies a partial solution, otherwise $a_e(\eta) = +\infty$. For a leaf link e of T, $D_e(\eta)$ is computed for every $\eta \in \{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^b$ by enumeration. For an internal link e of T, e has two child links e_1 and e_2. Let $b_1 = |\partial(A_{e_1})|$ and $b_2 = |\partial(A_{e_2})|$. The sets $D_e(\eta)$ are computed by combining the sets of $D_{e_1}(\eta_1)$ and the sets of $D_{e_2}(\eta_2)$, where η_1 is a coloring of $\{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^{b_1}$ and η_2 is a coloring of $\{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^{b_2}$.

Let $X_1 = \partial(A_e) \setminus \partial(A_{e_2})$, $X_2 = \partial(A_e) \setminus \partial(A_{e_1})$, $X_3 = \partial(A_e) \cap \partial(A_{e_1}) \cap \partial(A_{e_3})$, and $X_4 = (\partial(A_{e_1}) \cup \partial(A_{e_2})) \setminus \partial(A_e)$. Then $\partial(A_e) = X_1 \cup X_2 \cup X_3$, $\partial(A_{e_1}) = X_1 \cup X_3 \cup X_4$, and $\partial(A_{e_2}) = X_2 \cup X_3 \cup X_4$. A basic-coloring λ of $\partial(A_e)$ is formed from basic-colorings λ_1 of $\partial(A_{e_1})$ and basic colorings λ_2 of $\partial(A_{e_1})$ if:

1. For $u \in X_1$, $\lambda(u) = \lambda_1(u)$.
2. For $u \in X_2$, $\lambda(u) = \lambda_2(u)$.
3. For $u \in X_3$, if $\lambda_1(u) = \lambda_2(u) = 1$ then $\lambda(u) = 1$; if $\lambda_1(u) = \lambda_2(u) = \hat{0}$ then $\lambda(u) = \hat{0}$; and if $\lambda_1(u) = 0$ and $\lambda_2(u) = \hat{0}$, or $\lambda_1(u) = \hat{0}$ and $\lambda_2(u) = 0$ then $\lambda(u) = 0$.
4. For $u \in X_4$, $\lambda_1(u) = \lambda_2(u) = 1$, or $\lambda_1(u) = 0$ and $\lambda_2(u) = \hat{0}$, or $\lambda_1(u) = \hat{0}$ and $\lambda_2(u) = 0$.

For a basic-coloring λ which is formed by two basic-colorings λ_1 and λ_2, we compute the disjoint components $P_1, ..., P_r$ of $H[D_{e_1}(\eta_1) \cup D_{e_2}(\eta_2)]$, where for $i = 1, 2$ $\eta_i(u) = \lambda_i(u)$ if $\lambda_i(u) \in \{0, \hat{0}\}$ and $\eta_i(u) \in \{1_[, 1_], 1^*, \hat{1}\}$ if $\lambda_i(u) = 1$. $D_{e_1}(\eta_1) \cup D_{e_2}(\eta_2)$ is called a *candidate* for $D_e(\eta)$ if each P_i has at least one vertex in $\partial(A_e)$. If $D_{e_1}(\eta_1) \cup D_{e_2}(\eta_2)$ is a candidate, we convert the color of u with $\lambda(u) = 1$ into one color of $\{1_[, 1_], 1^*, \hat{1}\}$ according to if u is the small end, the

large end, a middle vertex, or the only vertex of $V(P_i) \cap \partial(A_e)$, respectively, to get a coloring $\eta \in \{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^b$. Finally, $D_e(\eta)$ is a candidate $D_{e_1}(\eta_1) \cup D_{e_2}(\eta_2)$ with the minimum cardinality.

The colorings $\{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^b$ and the corresponding partial solutions can be kept in a table of size $O(6^b)$. A bijection from $\{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^b$ to $\{1, 2, ..., 6^b\}$ gives an index method to access the table. The colorings $\{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^{b_1}$ and $\{0, \hat{0}, 1_[, 1_], 1^*, \hat{1}\}^{b_2}$ are handled similarly. The index method of DPBF Algorithm described above solves the CDS problem for a plane G of n vertices in $O(2^{4.67\mathrm{bw}(G)}\gamma_c(G) + n^3)$ time and $O(6^{\mathrm{bw}(G)}\gamma_c(G))$ memory space. The running time of the index method can be improved to $O(2^{4.618\mathrm{bw}(G)}\gamma_c(G) + n^3)$ by a more complex analysis for Step III. In Step III, merging colorings can be done by the distance matrix multiplication. If the conventional $O(n^3)$ time distance matrix multiplication is used, this gives the same running time as that of the index method. If the fast distance matrix multiplication is used, the running time of DPBF Algorithm can be further improved to $O(2^{9.8\sqrt{n}}\gamma_c(G) + n^3)$ and $O(2^{24.257\sqrt{\gamma_c(G)}}\gamma_c(G) + n^3)$. We omit the analysis of the above running times due to the limit of space.

4 Computational Results

We tested the performance of DPBF Algorithm on the following classes of planar graphs. Class (1) is a set of random maximal planar graphs and their subgraphs generated by LEDA [1,3]. Class(2) includes the Delaunay triangulations of point sets taken from TSPLIB [22]. The instances of Classes (3) and (4) are the triangulations and intersection graphs generated by LEDA, respectively. The instances of Class (5) are Gabriel graphs generated using the points uniformly distributed in a two-dimensional plane. The instances of Class (6) are random planar graphs generated by the PIGALE library [2].

We use the reduction rules of [16] to compute the kernels of input instances in Step I and the $O(n^3)$ time algorithm of [4] to compute optimal sphere-cut decompositions of kernels in Step II. For Step III, we use an index method to access the tables. To save memory, we compute the colorings of links of T in the postorder manner. Once the colorings of a link e are computed for a link e, the solutions for the children links of e are discarded. Because the fast distance matrix multiplication is not practical [21], applying this technique does not improve the practical performance of the algorithm.

The computer used for testing has an AMD Athlon(tm) 64 X2 Dual Core Processor 4600+ (2.4GHz) and 3GByte of internal memory. The operating system is SUSE Linux 10.2 and the programming language used is C++.

Table 1 shows the computational results of the simple version of DPBF Algorithm. H is the kernel of an instance computed in Step I. In Step II, an optimal sphere cut decomposition of H is computed and we report $|E(H)|$, the size of H, the branchwidth $\mathrm{bw}(H)$ of H, and the running time of this step. For Step III, we give $\gamma_c(G)$ obtained, the running time of the step and the required memory in Gigabytes (GB). All times in the table are in seconds.

Table 1. Computational results (time in seconds) of DPBF Algorithm. For the instances marked with "*", the 3GByte memory is not enough for computing a minimum connected dominating set.

Class	Graph	$\|E(G)\|$	$bw(G)$	Step I	Step II			Step III		total	maximum
	G			time	$\|E(H)\|$	bw(H)	time	$\gamma_c(G)$	time	time	memory
(1)	max1000	2912	4	19.4	704	4	2.3	131	4	25.7	
	max2000	5978	4	63	1133	4	6.0	252	9.9	78.9	
	max3000	8510	4	359	2531	4	37.6	417	94	491	
	max4000	10759	4	836	3965	4	145	614	458	1439	
	max5000	14311	4	848	3873	4	160	650	383	1392	
	max5000	16206	4	1702	5989	4	325	907	1769	3796	
(2)	eil51	140	8	0.1	140	8	0.2	14	253	254	0.03
	lin105	292	8	0.3	275	8	3	27	810	813	0.03
	pr144	393	9	1	347	7	0.5	25	18.1	19.7	0.06
	kroB150	436	10	1	436	10	0.8	36	133856	133858	1.05
	pr226	586	7	1.3	399	6	1.7	24	5.1	8.1	0.04
	ch130	377	10	0.3	377	10	0.6	34	38562	38563	0.74
(3)	tri100	288	7	0.7	258	6	0.6	20	7.1	8.4	0.05
	tri500	1470	7	10.1	1438	6	37.2	91	62.6	110	0.07
	tri800	2374	8	18	2279	7	86.4	149	289	393	0.13
	tri2000	5977	8	109	5751	8	603	369	5643	6355	0.48
	tri4000	11969	9	547	11236	9	3690	753	42323	46560	0.57
(4)	rand100	121	5	0.1	73	3	0.1	40	0.1	0.3	0.03
	rand500	709	7	1.7	545	6	0.4	216	10.8	12.9	0.05
	rand700	1037	7	2.9	836	6	1	301	17.8	21.8	0.07
	rand1000	1512	8	4.5	1242	7	2.5	421	422.8	429.8	0.25
	rand2000	3247	8	17.5	2852	8	17.8	839	10179	10214	0.38
	rand3000*	4943	10	-	-	10	-	-	-	-	
(5)	Gab50	88	4	0.1	88	4	0.1	22	0.2	0.4	0.03
	Gab100	182	7	0.1	179	7	0.3	41	66.7	67.1	0.11
	Gab200	366	8	0.7	362	8	1.5	81	2290	2293	0.13
	Gab300	552	10	1.4	545	10	1.6	121	12 days	12 days	2.53
(6)	P206	269	4	0.6	163	4	0.3	78	0.3	1.2	0.02
	P495	852	5	3.2	765	5	8.4	167	11.9	23.5	0.02
	P855	1434	6	7.9	1280	6	15.1	289	77.9	101	0.06
	P1000	1325	5	4.4	777	5	2.5	378	7.3	14.2	0.07
	P2000	2619	6	24.5	1527	6	12.3	738	58.0	94.8	0.11
	P4206	7101	6	256	6377	6	1816	1423	2411	4482	0.43

Now we go over the details of our results. It is shown in [21] that the branchwidth of the instances of class (1) is at most four. Our results show that reduction rules are very effective on these graphs and that the size of the kernels is much smaller than the size of the original graphs. Thus, Step III is fast and the minimum CDS of some instances with 16000 edges can be computed in about one hour on our platform.

However, the branchwidth increases very fast in the size of the graph for the instances of Classes (2) and (5). In addition, the reduction rules do not reduce the size of the original graphs very much, and the size and branchwidth of generated kernels are the same as those of the original graphs. The running time of Step III increases significantly with the branchwidth of instances (e.g., see the running time of instances pr144 and kroB150). For instances with the same branchwidth the running time of this step depends on the size of the kernel. (see instances eil51 and lin105). For these classes of planar graphs DPBF Algorithm is time consuming and can solve the CDS problem on instances of size up to a few hundreds edges in a practical time. The branchwidth of instances

of Classes (3) and (4) grows relatively slow in the instance sizes. Furthermore, data reduction rules are effective on the instances of Class (4). The branchwidth of graph instances in Class (6) does not grow in the instance size thus, DPBF Algorithm is efficient for this class.

The memory space required by DPBF Algorithm in Step III is a bottleneck for solving instances with large branchwidth. Experimental results show that DPBF Algorithm can compute a minimum CDS for instances with the branchwidth of kernels at most 10 ($\mathrm{bw}(H) \leq 10$) using 3GBytes of memory space.

5 Concluding Remarks

We evaluated the performance of DPBF Algorithm for the CDS problem on a wide range of planar graphs. The computational results coincide with the theoretical analysis of the algorithm, it is efficient for graphs with small branchwidth but may not be practical for graphs with large branchwidth. Using a computer with a CPU of 2.4GHz and 3GMBytes memory space, it is possible to find a minimum CDS for graphs with the branchwidth of their kernels at most 10 in a few hours. Since the branchwidth of a planar graph can be computed in $O(n^2 \log n)$ time by the $O(n^2)$ time rat-catching algorithm [26] and a binary search, one may first get the branchwidth of the input graph and then decide if DPBF Algorithm is applicable using the results of this paper as a guideline.

Because DPBF Algorithm runs and requires memory space exponentially in the branchwidth $\mathrm{bw}(H)$ of a kernel H for a given graph, it is worth to develop more powerful data reduction rules to reduce $\mathrm{bw}(H)$. It is known that the planar CDS problem admits PTAS [7]. The approach for the PTAS is to partition an input graph into subgraphs of fixed branchwidth, find a minimum CDS for each subgraph and combining the solutions of subgraphs into a solution of the input graph. It is interesting to apply DPBF Algorithm to develop PTAS which is efficient in practice for the planar CDS problem in graphs with large branchwidth.

References

1. Library of Efficient Data Types and Algorithms, Version 5.2 (2008), `http://www.algorithmic-solutions.com/enleda.htm`
2. Public Implementation of a Graph Algorithm Library and Editor (2008), `http://pigale.sourceforge.net/`
3. The LEDA User Manual, Algorithmic Solutions, Version 4.2.1 (2008), `http://www.mpi-inf.mpg.de/LEDA/MANUAL/MANUAL.html`
4. Bian, Z., Gu, Q.: Computing branch decompositions of large planar graphs. In: McGeoch, C.C. (ed.) WEA 2008. LNCS, vol. 5038, pp. 87–100. Springer, Heidelberg (2008)
5. Blum, J., Ding, M., Thaeler, A., Cheng, X.: Connected dominating set in sensor networks and MANETs. In: Du, D.-Z., Pardalos, P. (eds.) Handbooks of Combinatorial Optimization, Suppl., vol. B, pp. 329–369. Springer, Heidelberg (2004)
6. Cheng, X., Ding, M., Du, H., Jia, X.: Virtual backbone construction in multihop Ad Hoc wireless networks. Wireless Communications and Mobile Computing 6(2), 183–190 (2006)

7. Demaine, E.D., Hajiaghayi, M.: Bidimensionality: new connections between FPT algorithms and PTAS. In: Proc. of the 2005 ACM/SIAM Symposium on Discrete Algorithms (SODA 2005), pp. 590–601 (2005)
8. Dorn, F.: Dynamic programming and fast matrix multiplication. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 280–291. Springer, Heidelberg (2006)
9. Dorn, F., Penninkx, E., Bodlaender, H.L., Fomin, F.V.: Efficient exact algorithms on planar graphs: exploiting sphere cut branch decompositions. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 95–106. Springer, Heidelberg (2005)
10. Dorn, F., Penninkx, E., Bodlaender, H.L., Fomin, F.V.: Efficient exact algorithms on planar graphs: exploiting sphere cut decompositions. Technical report, UU-CS-2006-006, Department of Information and Computing Sciences (2006)
11. Downey, R.G., Fellow, M.R.: Parameterized complexity. In: Monographs in Computer Science. Springer, Heidelberg (1999)
12. Downey, R.G., Fellows, M.R.: Fixed parameter tractability and completeness. Cong. Num. 87, 161–187 (1992)
13. Fomin, F.V., Thilikos, D.M.: Dominating sets in planar graphs: branch-width and exponential speed-up. SIAM Journal on Computing 36(2), 281–309 (2006)
14. Fomin, F.V., Thilikos, D.M.: New upper bounds on the decomposability of planar graphs. Journal of Graph Theory 51(1), 53–81 (2006)
15. Garey, M.R., Johnson, D.S.: Computers and Intractability, a Guide to the Theory of NP-Completeness. Freeman, New York (1979)
16. Gu, Q., Imani, N.: Connectivity is not a limit for kernelization: planar connected dominating set. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 26–37. Springer, Heidelberg (2010)
17. Gu, Q., Tamaki, H.: Optimal branch-decomposition of planar graphs in $O(n^3)$ time. ACM Transactions on Algorithms 4(3), 30:1–30:13 (2008)
18. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmca 20, 374–387 (1998)
19. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Domination in graphs. In: Monographs and Textbooks in Pure and Applied Mathematics, vol. 209. Marcel Dekker, New York (1998)
20. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Fundamentals of domination in graphs. In: Monographs and Textbooks in Pure and Applied Mathematics, vol. 208. Marcel Dekker, New York (1998)
21. Marzban, M., Gu, Q., Jia, X.: Computational study on planar dominating set problem. Theoretical Computer Science 410(52), 5455–5466 (2009)
22. Reinelt, G.: TSPLIB-A traveling salesman library. ORSA J. on Computing 3, 376–384 (1991)
23. Robertson, N., Seymour, P.D.: Graph minors I. Excluding a forest. Journal of Combinatorial Theory, Series B 35, 39–61 (1983)
24. Robertson, N., Seymour, P.D.: Graph minors II. Algorithmic aspects of tree-width. Journal of Algorithms 7, 309–322 (1986)
25. Robertson, N., Seymour, P.D.: Graph minors X. Obstructions to tree decomposition. J. of Combinatorial Theory, Series B 52, 153–190 (1991)
26. Seymour, P.D., Thomas, R.: Call routing and the ratcatcher. Combinatorica 14(2), 217–241 (1994)