Volume 5, Issue 6, June 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

# Recognizing Dominating Graphs with σ ≥ n/3

**Hoa T. Thi***
Department of Natural Sciences
Nam Dinh Teacher Training Colleg, Vietnam

**Hoa V. Dinh**
Department of Information Technology,
HaNoi National University of Educationi, Vietnam

***Abstract:*** ***Given an undirected and simple graph G with n vertices. A cycle C is called a Hamilton cycle if C goes through all vertices of G. A cycle C is said to be dominating cycle if and only if G-C has no edge.***

***The problem DC of recognizing a dominating cycle in a given graph was proved to be a NPC-problem [3]. In this paper, we present a polynomial algorithm to estimate dominating cycle for the class of 2-connected graphs satisfying*** $\sigma(G) \geq \frac{n}{3}$***, where σ(G) is the minimum degree of vertices in G.***

***Keywords:*** ***dominating cycle, dominating graph, 2-connected graph, polynomial algorithm, 2-connected graph with minimal degree σ ≥ n/3.***

## I. INTRODUCTION

In this paper we use terminology and notation in [1]. Given an undirected and simple graph *G* with n vertices. A cycle *C* is called a Hamilton cycle if *C* goes through all vertices of *G* [8]. A cycle *C* in *G* is said to be dominating cycle if and only if *G-C* has no edge, and *G* will be called a cycle- dominable graph, or short, a dominating graph.

Given two graphs *G=(V,E)* and *G'=(V',E')*. The graphs *G* and *G'* are called disjoint when $V \cap V' = \Phi$. If $V \subseteq V'$ and $E \subseteq E'$, we write $G \subseteq G'$ and say that the graph *G'* is a subgraph of *G*. We denote by $G * G'$ the graph containing from the union of the graphs $G \cup G'$, the graph with the vertices set $V \cup V'$ and the set of edges $E \cup E'$, by connecting all vertices of *G* with any vertex of *G'*. Let $K_n$ denote the complete graph with n vertices. Let $s \geq 2$ and $n_1, n_2,....,n_s \geq 1$ then $\overline{K}_{n_1,n_2,...n_s}$ is the union of s disjoint complete graphs $K_{n_1}, K_{n_2}, ...K_{n_s}$. In case $n_1 = ... = n_s = n$ we use $\overline{K}_{s*n}$ instead of $\overline{K}_{n_1,n_2,...n_s}$.

The problem of determining the existence of a dominating cycle in the graph is stated as follows:

**Dominating cycle (DC)**

**Instance**: Give an undirected graph *G*.

**Question**: Does there exist a dominating cyle *C* in *G*?

Problem DC has been shown NP-complete [3], so we do not have a good algorithm (done in a polynomial time) to solve it, we consider it in a special class of graphs. A NP-complete problem may be a P-problem in a special class of graphs [4]. For the graphs with $\sigma(G) \geq \frac{n+2}{3}$ Nash-Williams [7] has proved the following theorem:

**Theorem 1** Let *G* be 2-connected graph with n vertices satisfying $\sigma(G) \geq \frac{n+2}{3}$. Then every longest cycle in *G* is a dominating cycle.

Bondy [2] generalized Theorem 1:

**Theorem 2** Let *G* be a 2-connected graph with n vertices satisfying $d(x)+d(y)+d(z) \geq n+2$ for all pair-wise non-adjacent vertices *x, y, z*. Then every longest cycle of *G* is a dominating cycle.

Studying the class of 2-connected graphs with $\sigma(G) \geq \frac{n}{3}$ Vu Dinh Hoa [5] has proposed and proved the following theorem:

**Theorem 3** Let *G* be a 2-connected graph with $\sigma(G) \geq \frac{n}{3}$ and *C* a longest cycle of *G*. Then *G-C* is eitheir a edgesless graph or a complete graph.

## II. RESULTS

According to [5] we have the following theorem:

**Theorem 4** Let *G* be a 2-connected graph satisfying $\sigma(G) \geq \frac{n}{3}$. Then the following properties are equivalent:

 

(a) $G$ is a cycle- dominable graph.

(b) Every longest cycle in $G$ is a dominating cycle.

(c) $G \notin K$ for a class $K$ of special graphs.

The class $K$ is the union of the classes $K_1, K_2, K_3, K_4, K_5$ described as follows.

$K_1$ is the class of graphs of 3r vertices with minimal degree $\sigma = r \geq 3$, which is isomorphic to a graph $G$ with $\overline{K}_{r-1,r-1} * \overline{K}_2 \subseteq G \subseteq \overline{K}_{r-1,r-1,r} * K_2$;

$K_2$ is the class of graphs of 15 vertices with minimal degree $\sigma = 5$, which is isomorphic to a graph $G$ with $\overline{K}_{3,3,3,3} * \overline{K}_3 \subseteq G \subseteq \overline{K}_{3,3,3,3} * K_3$;

$K_3$ is the class of graphs of 3r vertices with minimal degree $\sigma = r \geq 3$, which is isomorphic to graph $G$ with $\overline{K}_{(r-1)*2} * \overline{K}_{r-1} \subseteq G \subseteq \overline{K}_{(r-1)*2,3} * K_{r-1}$;

$K_4$ is the class of graphs of 3r+2 vertices and with minimal degree $\sigma = r+1 \geq 3$ ,which is isomorphic to graph $G$ with $\overline{K}_{3*r} * \overline{K}_2 \subseteq G \subseteq \overline{K}_{3*r} * K_2$;

$K_5$ is the class of graphs of $\sigma = r+1 \geq 3$ vertices and with minimal degree $\sigma = r \geq 3$, which is isomorphic to graph $G$ with $\overline{K}_{(r+1)*2} * \overline{K}_r \subseteq G \subseteq \overline{K}_{(r+1)*2} * K_r$.

In [6], the problem of isomorphic subgraph is stated in the form of the decisive problem:

**Isomorphic subgraph**

**Instance**: Graphs $G_1$ and $G_2$.

**Question**: Whether graph $G_1$ is isomorphic to one of the subgraphs of $G_2$?

Note that the isomorphic subgraph problem is proven to be NP-complete [6]. We now state the problem determining whether a given graph $G$ is isomorphic to graph $K_i$ *(i=1..5)* as the decisive problem as follows:

**Isomorphic G_K$_i$** (i=1..5)

**Instance**: Graph $G$.

**Question**: Whether graph $G$ is isomorphic to one of the subgraphs of $K_i$.

Based on Theorem 4, we give algorithms determine the existence of a dominating cycle in a 2-connected graph satisfying the minimum of degree $\sigma(G) \geq \frac{n}{3}$ and show that the problems *Isomorphic G_K$_i$* are still in class *P*.

***A. Idea***

When $G$ is a 2-connected graph with $\sigma(G) \geq \frac{n}{3}$, we can decide the existence of a dominating cycle $C$ by determining if the graph $G$ belongs to a special class of graphs $K$. If $G \notin K$ , $G$ will contain a dominating cycle. Otherwise, $G$ has no dominating cycles.

In order to check whether graph $G$ belongs to a class $K$, we must prove that $G$ is a subgraph of a graph in $K$. If $G$ is not a subgraph of a graph in $K$ , we conclude that $G$ does not belongs to $K$; otherwise, $G$ belongs to $K$.

**Algorithm Dominating-Test**

**Input**: $G$ is an undirected graph

**Output**: $G$ contains a dominating cycle or $G$ does not contain a dominating cycle.

**Methods**:

I. If $\sigma < \frac{n}{3}$, stop with NO.

II. Check whether $G$ is 2-connected graph. If not, stop with NO.

III. If $(G \in K_1)$ or $(G \in K_2)$ or $(G \in K_3)$ or $(G \in K_4)$ or $(G \in K_5)$ , $G \in K$ and $G$ does not contain a dominating cycle. If not, $G \notin K$ and $G$ contains a dominating cycle.

**Details of Step II:** {Whether $G$ is 2-connected graph}

$G$ is a 2-connected graph if and only if we remove any vertex and edges of it, the remaining graph is connected.

**Algorithm 2-Connected-Test**

1. If $G$ is not connected graph, stop with NO. If not, go to step 2.
2. From any vertex i, mark i and go to step 3.
3. Check if ($G$-i) is connected graph, go to step 4. If not, stop with NO.
4. Repeat step 2 until it does not work, then go to step 5.
5. Check if number of marked vertices is < $n$, $G$ is not 2-connected. If not, $G$ is 2-connected.

To check whether $G$ is connected graph, we need $O(n^2)$ time. For each vertex of graph $G$, we mark it to be deleted, then we check whether remaining graph is connected. So the algorithm 2-Connected-Test requires $O(n^3)$ time.

**Details of Step III:** {Check the dependence of graph $G$ on the class of graphs $K_1$ , $K_2$ , $K_3$ , $K_4$ , $K_5$}.

Symbols are used in the algorithms:

- $V$: set of vertices of graph $G$.
- $|V|$*:* the number of elements of set $V$.

 

- $K_n$: complete graph $K$ with $n$ vertices.
- *a[i,j]=0*: vertex *i* does not connect to vertex *j; a[i,j]=1*: vertex *i* connects to vertex *j*.

**Algorithm K-1-Test** $\{ \overline{K}_{r-1,r-1} * \overline{K}_2 \subseteq G \subseteq \overline{K}_{r-1,r-1,r} * K_2 \}$

**Input**: Graph *G* and the class of graph $K_1$.
**Output**: Whether *G* is isomorphic to one of subgraphs of $K_1$.
**Method:**

1. If |*V*| <> *3r*, stop with NO.
2. Create set *S* containing vertices which degree of them ≥ *2(r-1)*. If |*S*|<*2*, stop with NO.
3. Check whether set *G*-S contains *2(r-1)* complete graphs $K_{r-1}$? If not, stop with NO.
4. Check whether two vertices in *S* are connected to all vertices of two complete graphs $K_{r-1}$? If not, stop with NO.
5. Stop with YES {$G \in K_1$}.

**Details of Step 3:**

a) Let all initial vertices of *G* be not deleted: *fix[i]=0*. For each vertex *j* in *S*, let *j* is deleted: *fix[j]=1*. We use var *count* to count number of complete graphs $K_{r-1}$.

b) For each vertex i in set *G*-S:

- Create array *K* which contains vertex i and its neighbor.
- If *K* is complete graph with *r-1* vertices, we increase var *count* and mark vertices of *K* as *fix[j]=2*.

c) If *count<2*, stop with NO.

**Details of Step 4:**

a) We use var *ds* to count number of vertices of set S which connect to all vertices of two complete graphs $K_{r-1}$, let ds=0.

b) For each vertex *i* in *S*

- Create var *d* to count number of vertices which connect to vertex *i*;
- For each vertex *j* in *G-S*, if *fix[j]=2* and *a[i,j]=1*, *d=d+1*;
- If *d* ≥ *2(r-1)*, *ds=ds+1*

c) If *d <2*, stop with NO.

Step *1* requires *O(1)* time. For each vertex i in V, if i has degree ≥ *2(r-1)*, add vertex *i* to set *S*. So step *2* requires *O(n)* time. In step *3*, for each vertex i we have to check which other vertices are adjacent to it and whether I and this set of vertices make up the complete graph or not. Then we need to use the test function to see whether the graph is a complete graph (a complete graph with n vertices is the graph which the degree of every vertex equals *n-1*). This function requires $O(n^3)$ time. So step *3* requires $O(n^3)$ time. Step *4* requires $O(n^2)$ time. Thus, the overall time required by algrorithm K-1-Test is $O(n^3)$.

**Algorithm K-2-Test** $\{ \overline{K}_{3,3,3,3} * \overline{K}_3 \subseteq G \subseteq \overline{K}_{3,3,3,3} * K_3 \}$

**Input**: Graph *G* and the class of graph $K_2$.
**Output**: Whether *G* is isomorphic to one of subgraphs of $K_2$.
**Method:**

1. If |*V*| <> *15*, stop with NO.
2. Searching three u, v and w which have maximal degree ≥ 12. If not, stop with NO.
3. Check whether set *G-{u,v,w}* contains four complete graphs $K_3$? If not, stop with NO.
4. Check whether every vertex in $K_{3,3,3,3}$ connects to three vertices *u, v* and *w*? If not, stop with NO.
5. Stop with YES {$G \in K_2$}.

**Details of Step 3:**

a) Let all initial vertices of *G* be not deleted: *fix[i]=0* and let vertices *u, v* and *w* be deleted: *fix[u]=1, fix[v]=1, fix[w]=1*. We use var *count* to count number of complete graph $K_3$.

b) For each vertex *i* in set *G-{u,v,w}*:

- Create array *K* which contains vertex *i* and its neighbor.
- If *K* is complete graph with *3* vertices, we increase var *count* and mark vertices of *K* as *fix[j]=2*.

c) If *count <>4*, stop with NO.

**Details of Step 4:**

- For each vertex j in *G-{u,v,w}* and *fix[j]=2*, if *a[u,j]=0* or *a[v,j]=0* or *a[w,j]=0*, stop with NO.

The same as evaluating the complexity of algorithm *K-1-Test* , algorithm *K-2-Test* requires *O(n3)* time.

**Algorithm K-3-Test** $\{ \overline{K}_{(r-1)*2} * \overline{K}_{r-1} \subseteq G \subseteq \overline{K}_{(r-1)*2,3} * K_{r-1} \}$

**Input**: Graph *G* and the class of graph $K_3$.
**Output**: Whether *G* is isomorphic to one of subgraphs of $K_3$.
**Method:**

1. If |*V*| <> *3r*, stop with NO.
2. Create set *S* containing vertices which degree of them ≥ *2(r-1)*. If |*S*| ≤ *r-1* stop with NO.
3. Check whether set *G*-S contains *(r-1)* complete graphs $K_2$? If not, stop with NO.
4. Check whether *r-1* vertices in *S* are connected to all vertices of *r-1* complete graphs $K_2$ ? If not, stop with NO.
5. Stop with YES {$G \notin K_3$}.

 

**Details of Step 3:**

a) Let all initial vertices of *G* be not deleted: *fix[i]=0*. For each vertex *j* in *S*, let j is deleted: *fix[j]=1*. We use var *count* to count number of complete graph $K_2$.

b) For each vertex *i* in set *G*-S:

- Create array *K* which contains vertex *i* and its neighbor.
- If K is complete graph with *2* vertices, we increase var *count* and mark vertices of *K* as *fix[j]=2*.

c) If *count<r-1*, stop with NO.

**Details of Step 4:**

a) We use var *ds* to count number of vertices of set *S* which connect to all vertices of *r-1* complete graphs $K_2$, let *ds=0*.

b) For each vertex *i* in *S*

- Create var *d* to count number of vertices which connect to vertex *i*;
- For each vertex *j* in *G*-S, if *fix[j]=2* and *a[i,j]=1*, *d=d+1*;
- If $d \geq 2(r-1)$, *ds=ds+1*

c) `If *ds<r-1*, stop with NO.

Algorithm K-3-Test requires $O(n3)$ time.

**Algorithm K-4-Test** $\{ \overline{K}_{3*r} * \overline{K}_2 \subseteq G \subseteq \overline{K}_{3*r} * K_2 \}$

**Input**: Graph *G* and the class of graph $K_4$.

**Output**: Whether *G* is isomorphic to one of subgraphs of $K_4$.

**Method:**

1. If *|V| <> 3r+2*, stop with NO.
2. Searching two vertices *u* and *v* which degree of them ≥ *3r*. If not, stop with NO.
3. Check whether set *G*-{u,v} contains four complete graphs $K_r$? If not, stop with NO.
4. Check whether two vertices *u* and *v* connect to all vertices of three graphs $K_r$? If not, stop with NO.
5. Stop with YES $\{ G \in K_4 \}$.

**Details of Step 3:**

d) Let all initial vertices of *G* be not deleted: *fix[i]=0* and let vertices *u* and *v* be deleted: *fix[u]=1*, *fix[v]=1*. We use var *count* to count number of complete graph $K_r$.

e) For each vertex *i* in set *G*-{*u,v*}:

- Create array *K* which contains vertex *i* and its neighbor.
- If *K* is complete graph with *r* vertices, we increase var *count* and mark vertices of *K* as *fix[j]=2*.

f) If *count <>3,* stop with NO.

**Details of Step 4:**

- For each vertex *j* in *G*-{*u,v*} and *fix[j]=2*, if *a[u,j]=0* or *a[v,j]=0*, stop with NO.

Algorithm K-4-Test requires $O(n^3)$ time.

**Algorithm K-5-Test** $\{ \overline{K}_{(r+1)*2} * \overline{K}_r \subseteq G \subseteq \overline{K}_{(r+1)*2} * K_r \}$

**Input**: Graph *G* and the class of graph $K_5$.

**Output**: Whether *G* is isomorphic to one of subgraphs of $K_5$.

**Method:**

1. If *|V| <> 3r+2*, stop with NO.
2. Create set *S* containing vertices which have degree ≥ *2(r+1)*. If *|S|<r*, stop with NO.
3. Check whether set *G*-S contains *r+1* complete graphs $K_2$? If not, stop with NO.
4. Check whether *r* vertices in *S* are connected to all vertices of *r+1* complete graphs $K_2$? If not, stop with NO.
5. Stop with YES $\{G \in K_5\}$.

**Details of Step 3:**

a) Let all initial vertices of *G* be not deleted: *fix[i]=0*. For each vertex *j* in *S*, let *j* is deleted: *fix[j]=1*. We use var *count* to count number of complete graph $K_2$.

b) For each vertex *i* in set *G*-S:

- Create array *K* which contains vertex *i* and its neighbor.
- If *K* is complete graph with *2* vertices, we increase var *count* and mark vertices of *K* as *fix[j]=2*.

c) If *count<r+1*, stop with NO.

**Details of Step 4:**

d) We use var *ds* to count number of vertices of set S which connect to all vertices of *r+1* complete graphs $K_2$, let *ds=0.*

e) For each vertex *i* in *S*

- Create var *d* to count number of vertices connecting to vertex *i*;
- For each vertex *j* in *G*-S, if *fix[j]=2* and *a[i,j]=1*, *d=d+1*;
- If d ≥ 2(r+1), *ds=ds+1*

f) If *ds<r+1*, stop with NO.

Algorithm K-5-Test requires $O(n^3)$ time.

 

***B. Evaluation of complex algorithms***

*1) Proof of correcteness*

The algorithm which determines the existence of a dominating cycle in 2-connected graph and satisfies $\sigma \geq \frac{n}{3}$ is true.

We examine the algorithm which determines the dependence of the graph $G$ on graph $K_1$ as follows:

Graph $G$ contains two vertices which have degree $2(r-1)$, the degree of remaining vertices $\leq (r-1)+2=r+1$. We see that $2(r-1) \geq r+1$, so we examine the case:

- $r=3$: $G$ contains $\geq 2$ vertices which have degree $\geq 2(r-1)$.
- $r>3$: $G$ contains only 2 vertices which have degree $\geq 2(r-1)$.

So, we create set $S$ to contain vertices which has degree $\geq 2(r-1)$ and check whether set $G-S$ contains $\geq 2$ complete graphs $K_{r-1}$. If yes, check whether two vertices in set S connect to all vertices of two $K_{r-1}$ mentioned above. If yes, $G$ in $K_1$; otherwise, $G$ is not in $K_1$.

The algorithms for $K_2, K3, K_4, K_5$ are similar to the algorithm for $K_1$.

*2) The complecxity of Algorithm Dominating-Test*

Step I requires $O(1)$ time. We use algorithm 2-Connected-Test in step II, so step II requires $O(n^3)$ time. The algorithms which determine the dependence of the graph $G$ with each subgraph class of graph $K$ take $O(n^3)$ operations. So, step III requires $O(n^3)$ time.

Thus, the overall time required by algorithm Dominating-Test is $O(n^3)$.

## III. CONCLUSIONS

Based on the result of Theorem 4, we have developed the algorithm which determines the existence of dominating cycle in the graph $G$ by identifying the dependence of $G$ on the special class K. This algorithm has the complexity $O(n^3)$ and it is a polynomial time algorithm.

## REFERENCES

[1] R. Diestel , *Graph Theory*, Second Edition, Springer, 2000.

[2] Hao Li, Shan Zhou and Guanghui Wang, *The k – Dominating cycles in graphs*, Unite' Mixte de Recherche 8623 CNRS-Universte' Paris Sud-LRI, 2007.

[3] V. D. Hoa and D. N. An, "Recognizing dominating cycles is NP-hard*", Tạp chí tin học và điều khiển học*, T18, S. 3, 223-227, 2002.

[4] V. D. Hoa and D. N. An, "Necessary and sufficient condition for Maximal uniquely Hamiltonian graph", *International Journal Of Advanced Research In Computer Science*, pp. 114-117, 2012.

[5] Vũ Đình Hòa, 1986. *"*Định lí về cấu trúc đồ thị với số đủ lớn các cạnh", Hà Nội, tr. 519-532, 1986.

[6] R.Garey Michael and S.JohnSon David, *Computers and intractability*. A guide to the Theory of NP-Completeness, Spring Verlag, 1978.

[7] *C*. St. J. A. Nash-Williams, "Edge-disjoint Hamiltonian circuits in graphs with vertices of large valency*":*, in: L. Mirsky, ed., *Studies in Pure Mathematics*, Academic Press, London, pp. 157-183, 1971.

[8] N. H. X. Truong *and V. D. Hoa, "*Hamiltonian cycle in graphs $\sigma_4 \geq 2n$", *International Journal of Computer Science and Business Informatics*, Vol. 15, No. 2, pp. 38-60, 2015.

 