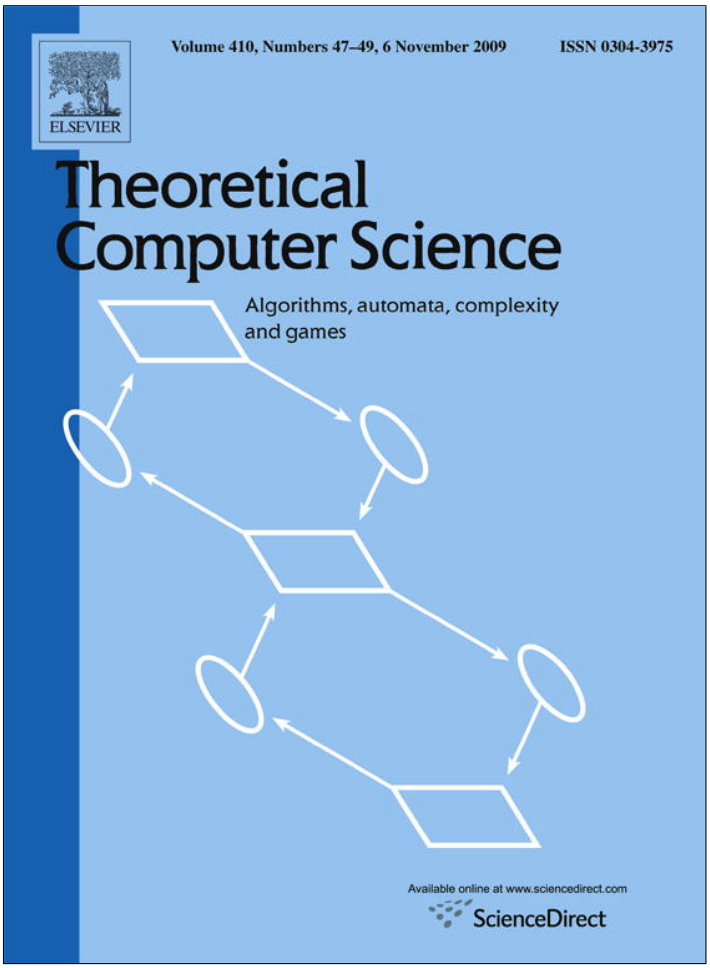

Theoretical Computer Science 410 (2009) 5122–5127

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

A randomized algorithm for determining dominating sets in graphs of maximum degree five

Soheir M. Khamis [a,*], Sameh S. Daoud [a], Hanaa A.E. Essa [b]

[a] Division of Computer Science, Department of Math., Faculty of Science, Ain Shams University, Cairo, Egypt

[b] Department of Math., Faculty of Science, Tanta University, Tanta, Egypt

ARTICLE INFO

Article history:
Received 28 September 2008
Received in revised form 22 June 2009
Accepted 17 August 2009
Communicated by J. Diaz

Keywords:
Minimum dominating set
Las Vegas technique
Randomized algorithm
Polynomial-time approximation algorithm

ABSTRACT

The paper is devoted to demonstrating a randomized algorithm for determining a dominating set in a given graph having a maximum degree of five. The algorithm follows the Las Vegas technique. Furthermore, the concept of a 2-separated collection of subsets of vertices in graphs is used. The suggested algorithm is based on a condition of the upper bound of the cardinality of a local dominating set. If the condition is not satisfied, then the algorithm halts with an appropriate message. Otherwise, the algorithm determines the dominating set. The given algorithm is considered a polynomial-time approximation one.

1. Introduction

This paper deals with the *Minimum Dominating Set, MDS*, problem in special graphs. Here, special graphs under consideration are those graphs having a maximum degree of five. A subset $\mathbb{D}$ of vertices in a graph G is called *a Dominating Set*, *DS*, of G if every vertex in G is either contained in $\mathbb{D}$ or adjacent to a vertex in $\mathbb{D}$.

If a graph G is connected, then a Connected Dominating Set, *CDS*, is defined as a *DS* which induces a connected subgraph in G. Computing a *Minimum CDS*, *MCDS*, can be considered as finding a spanning tree with a maximum number of leaves in a graph as mentioned in [8]. Determining a *MCDS* is one of the special problems of the domination set problem in graphs; other special problems are presented in [3]. In general, the *MDS* problem in graphs is *NP*-complete, [1]. The simple algorithm for determining a *MDS* in any arbitrary graph with n vertices, which enumerates and checks all subsets of vertices in graphs, takes $O(2^n)$ time. In [4], exact exponential algorithms for the *DS* problem which have a time of less than $O(2^n)$ are found, e.g., in arbitrary graphs ($O(1.93782^n)$), graphs of a maximum degree of three ($O(1.64515^n)$), and bipartite graphs ($O(1.73206^n)$). The *MDS* problem can be formulated as the covering set problem. This formulization leads to the fact that *the MDS problem can be solved in* $O(1.8021^n)$ *time*, [6]. The *MDS* problem in trees has a polynomial-time algorithm, [3]. In addition, some special problems of domination are solved in polynomial time for special graphs such as interval graphs, [3].

The paper is organized into five sections. In the next section, some basic definitions and concepts needed for the description of the topic are introduced. The complexity of the domination problem for the graphs having maximum degree five is given in Section 3. In Section 4, the description of a polynomial-time randomized algorithm for the *MDS* problem in 5-degree graphs and some results of computational experiments are presented. Finally, Section 5 includes concluding remarks about the introduced work of the paper.

* Corresponding author. Tel.: +20 022747034.
E-mail addresses: soheir_khamis@hotmail.com (S.M. Khamis), sameh_daoud2003@yahoo.com (S.S. Daoud), essahanaa@yahoo.com (H.A.E. Essa).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.08.011

2. Basic definitions and concepts

Let $G = (V, E)$ be an undirected simple graph, i.e., without loops and without multiple edges. A graph having maximum degree five is referred by a *5-degree graph*. For $V' \subseteq V$, $G[V']$ denotes the subgraph of G *induced* by V'. The vertex-set of $G[V']$ is V' and the edge-set consists of those edges of G with both end points in V'.

The *open neighborhood* of a vertex $v \in V$ is denoted by $N(v) = \{u \in V : \{u, v\} \in E\}$ and the *closed neighborhood or simply neighborhood* of $v$is denoted by $N[v] = N(v) \cup \{v\}$. The *degree of a vertex v* equals the cardinality of $N(v)$. The *neighborhood of a set* $S \subseteq V$ is defined as $N[S] = \cup_{s \in S} N[s]$. For $r \in N$, the rth *neighborhood* of $v \in V$ is defined recursively as $N_r[v] = N[N_{r-1}[v]]$, where $N_1[v] = N[v]$.

The *distance* $d(v, w)$ between two vertices $v, w \in V$ is the number of edges (or hops) that must be traversed to go from v to w on a shortest path. Similarly, the distance between two sets $A, B \subseteq V$ is defined as the distance between two closest elements

$a \in A$ and $b \in B$, i.e., $d(A, B) = \min\{d(x, y) : x \in A, y \in B\}$, [7]. Two sets $S, T \subseteq V$ are called 2-*separated* if and only if $d(S, T) \geq 3$, [7].

A set $D \subseteq V$ of a simple graph G is called a *dominating set* if every vertex $v \in V \backslash D$ is adjacent to some vertex $u \in D$. The *domination number* of the graph G which is usually denoted by $\gamma(G)$, is the cardinality of a smallest dominating set of G and such set is called *a minimum dominating set of* G. The dominating set problem is to determine $\gamma(G)$ and to find a dominating set of minimum cardinality.

Let $P(V)$ be the set of all subsets of vertices belonging to V. Let $D : P(V) \to P(V)$ be a function defined by: for every $W \in P(V)$, $D(W)$ is a dominating set having minimum cardinality in $G[W]$ and thus $D(W) \subseteq W$.

Now, we introduce the definition of the Las Vegas randomized approximation algorithm for an optimization problem. The terms which are used in the description of an optimization problem U are defined as follows, [2].

Let Σ_I and Σ_O be the set of input and output alphabets of U, respectively. The language of feasible problem instances is denoted by $L \subseteq \Sigma_I^*$, (all strings over Σ_I with any length k, $k \geq 0$). $L_I \subseteq L$ is the language of the (actual) problem instances of U. M is a function from L to P (Σ_O^*) and for every $x \in L$, $M(x)$ is called the set of feasible solutions for x. For every pair (u, x), cost is the cost function that assigns a positive real number cost (u, x), where $u \in M(x)$ for some $x \in L$. By goal, we mean either minimum or maximum.

Definition 2.1. Let $U = (\Sigma_I, \Sigma_O, L, L_I, M, \text{cost}, \text{goal})$ be an optimization problem. For any positive real $\delta > 1$, a randomized algorithm A is called a Las Vegas randomized δ-approximation algorithm for U if

(i) $Prob(A(x) \in M(x)) \geq 1/2$.
(ii) $Prob(R_A(x) \leq \delta) \geq 1/2$.

In Definition 2.1, condition (i) means that for every run of A, it computes a feasible solution of U with the probability at least $1/2$. In addition, condition (ii) states that a feasible solution, whose approximation ratio $R_A(x)$ is at most δ, is produced with a probability of at least $1/2$.

Dominating sets of subgraphs in 5-degree graphs

In what follows, we discuss the main idea of the suggested algorithm that depends on the concept of a 2-separated collection of subsets. The subgraphs induced by the subsets of such a collection divide the original graph into small parts for which it becomes easier to tackle the *MDS* problem. First, the basic property of a 2- separated collection which leads to obtain bound on the cardinality with respect to an optimal solution of dominating sets in graphs, is given as follows.

For any graph $G = (V, E)$, let $S = \{S_1, S_2, \ldots, S_k\}$ be a collection of subsets of vertices in G, i.e., $S_i \subset V$, for $i = 1, 2, \ldots, k$, satisfying the following property:

Property A. *For any two vertices $s \in S_i$ and $s' \in S_j$ with $i \neq j$, $d(s, s') > 2$. Then, S is known as a 2-separated collection of subsets of vertices in a graph, [5]. The following two lemmas are essential for designing the algorithm that will be given in Section 4.*

Lemma 2.1 ([5]). *For a 2-separated collection $S = \{S_1, S_2, \ldots, S_k\}$ in a graph $G = (V, E)$, we have*

$$|D(V)| \geq \sum_{i=1}^{k} |D(S_i)|.$$

Lemma 2.1 states that a 2-separated collection S leads to a lower bound on the cardinality of a *MDS* in G. Moreover, such a collection leads to finding an approximation of this cardinality. The next Lemma gives an approximation of *a MDS* in any graph G.

Lemma 2.2 ([5]). *Let $S = \{S_1, S_2, \ldots, S_k\}$ be a 2-separated collection in $G = (V, E)$. Given $\{T_1, T_2, \ldots, T_k\}$ is a collection in G with $\cup_{i=1}^{k} T_i = V$ satisfying $S_i \subset T_i$, for every $i = 1, 2, \ldots, k$. If there exists a real number $\rho \geq 1$ such that $|D(T_i)| \leq \rho \cdot |D(S_i)|$ holds for all $i = 1, 2, \ldots, k$, and $\cup_{i=1}^{k} D(T_i)$ forms a dominating set in G, then the set $\cup_{i=1}^{k} D(T_i)$ is a ρ- approximation of a MDS in G.*

3. The complexity of the domination problem in 5-degree graphs

In this section, it is shown that the dominating set problem in 5-degree graphs is *NP*-complete.

$\tilde{5}$-**SAT**:

Let $U = \{u_1, u_2, \ldots, u_n\}$ be a set of n boolean variables and $C = \{c_1, c_2, \ldots, c_m\}$ be a set of m clauses. C contains exactly one clause having five literals and the rest having at most five. The number of occurrences for each literal in the set of clauses is at most four, e.g., if $j \neq k \neq p \neq q$ and $u_i \in c_j, c_k, c_p$, and c_q, there exists no $z \in \{1, 2, \ldots, m\} - \{j, k, p, q\}$ such that $u_i \in c_z, \forall\, j, k, p, q \in \{1, 2, \ldots, m\}$.

The following theorem shows the complexity of $\tilde{5}$-*SAT* problem.

Theorem 3.1. *$\tilde{5}$-SAT is NP-complete.*

Proof. It is straightforward to show that $\tilde{5}$-*SAT* is *NP*-complete as for any *k-SAT*; $k \geq 3$, [1]. ■

Theorem 3.2. *The DS problem in 5-degree graphs is NP- complete.*

Proof. To prove the theorem, first, it is clear that the dominating set problem $\in$ *NP*. Then, we verify in polynomial time whether S is a dominating set in G or not. Next, a transformation from the $\tilde{5}$-*SAT* problem to the *DS* problem will be given. Assume that an instance, (U, C), of $\tilde{5}$-*SAT* is given. Then, an instance $G(C)$ of *DS* is built as follows. Construct an edge-set E' by joining u_i and $\bar{u}_i$ for each variable $u_i \in U$. For each clause $c_j \in C$, create a single vertex labeled c_j and add new edges to connect c_j with each literal in it. We will show that C is a 'yes' instance of $\tilde{5}$-*SAT* if and only if $G(C)$ is a 'yes' instance of *DS* for $k = n$ (the specified case). Suppose first that C has a satisfying truth assignment. Create a set S of vertices in $G(C)$ by testing: if $u_i =$ True, then put vertex u_i in S, otherwise, put $\bar{u}_i$ in S. The set S will be a dominating set of $G(C)$, since

(i) Each edge belonging to E' is incident to exactly one vertex in S. Therefore, for each edge, one of its vertices is either in S or is dominated by a vertex in S.
(ii) Each clause vertex c_j is dominated by at least one vertex in S.

Since by assumption, each clause contains at least one variable whose value is true. So, the corresponding vertex for the truth variable belongs to S. Therefore, $G(C)$ has a dominating set S of cardinality n. Conversely, suppose that $G(C)$ has a dominating set S of cardinality $\leq n$, and then we prove that C has a satisfying truth assignment. Clearly, each vertex of the form u_i must either be in S or be dominated by a vertex in S. In fact, each edge belonging to E' must be incident to exactly one vertex in S. Thus, S contains no clause vertex c_j. Each clause vertex is dominated by at least one vertex in S since S is a dominating set. Now, create a satisfying truth assignment for C by assigning the value *True* for any $u_i \in U$ if $u_i \in S$. Otherwise, assign to u_i the value *False*. It is straightforward to see that this is a satisfying truth assignment for C.

Finally, we show that creating an instance of *DS* from an instance of $\tilde{5}$-*SAT* carries out in a polynomial time. Evidently, the number of variables and clauses of an instance $\tilde{5}$-*SAT* is $O(5m + n)$, that is, C is specified by m sets of size at most five plus n variables. The graph $G(C)$ has $2n + m$ vertices and at most $n + 5m$ edges. The number of vertices and edges in $G(C)$ (cardinality of $G(C)$) is at most $\lambda \cdot |C|$, where λ is a constant greater than 0. Therefore, the graph $G(C)$ can be constructed from an instance of $\tilde{5}$-*SAT* in a polynomial time. ■

In the following section, Lemma 2.2 will be used with $\rho = 1.25$. We focus on the construction of suitable subsets $T_i \subset V$, which composes a 2-separated collection $S_i \subset T_i$ for $1 \leq i \leq |V|$. Furthermore, the other properties in the Lemma 2.2 are satisfied.

4. A randomized algorithm for determining a DS in 5-degree graphs

In this section, we will describe an algorithm to construct a dominating set of size at most $(1.25) \cdot |D(V)|$, where V is the set of vertices of G. An approximation solution is constructed by computing optimal dominating sets for small parts of the given graph and then taking the union of them. The algorithm works as follows. Initially, let a set $D = \Phi$, the algorithm chooses an arbitrary vertex $v \in V$and finds the rth neighborhood of v, for $r > 0$, where $N_0[v] = \{v\}$. Then, compute dominating sets of minimum cardinality for these neighborhoods as long as the following inequality (4.1) holds for all r.

$$|D(N_{r+2}[v])| > (1.25) \cdot |D(N_r[v])|. \tag{4.1}$$

Denote by $\hat{r}$ the smallest r at which (4.1) is violated. The value of r is increased until we find a corresponding $\hat{r}$. In this case, add $D(N_{\hat{r}+2}[v])$ to the current solution D. After that, remove all vertices in $N_{\hat{r}+2}[v]$ and edges related to them from the graph G. Let V' be the set of remaining vertices. The process repeats again for the remaining part of G by choosing a new vertex $v \in V'$, until a new $\hat{r}$ is found. The algorithm terminates when $V' = \Phi$ or $|D(N_{r+2}[v])| > ((r + 2) \cdot \Delta^{\frac{1}{2}} + 1)$ or $|D(N_r[v])| > (r \cdot \Delta^{\frac{1}{2}} + 1)$, where Δ is the maximum degree of the given graph. The algorithm is described formally in the following *RDS* -algorithm.

Algorithm RDS: //Compute a (1.25) -approximate *MDS* in a 5-degree graph.
Input: A graph $G = (V, E)$//G is a 5-degree graph.
1. $\mathbb{D} \leftarrow \Phi$;
2. $i \leftarrow l$;
3. **While** $V \neq \Phi$ **do**
4. Choose any $v \in V$;
5. $r \leftarrow 0$;
6. **While** $(|D(N_{r+2}[v])| > (1.25)*|D(N_r[v])|)$ **do**
7. **if** $(|D(N_{r+2}[v])| > (r+2)*\Delta^{\frac{1}{2}} + 1)$ or $|D(N_r[v])| > (r*\Delta^{\frac{1}{2}} + 1)$ **then** end the algorithm without solution;
8. **fi** ;
9. $r \leftarrow r + 1$;
10. **od**
11. $\mathbb{D} \leftarrow \mathbb{D} \cup D(N_{r+2}[v])$;
12. $V \leftarrow V/N_{r+2}[v]$;
13. $S_i \leftarrow N_r[v]$;
14. $T_i \leftarrow N_{r+2}[v]$;
15. $i \leftarrow i + 1$;
16. **od**
17. **return** $\mathbb{D}$.

It is clear that the algorithm *RDS* is terminated since the number of iterations, k, of the outer while-loop (steps 3–16) is bounded by $n = |V|$. Also, the output of the algorithm is the dominating set because all vertices of the graph are removed, i.e., all vertices are dominated. The algorithm terminates without an output when the condition in step 7 is satisfied.

The following lemma shows that the collection $\{S_1, S_2, \ldots, S_k\}$, which is constructed by the algorithm, is a 2-separated collection of subsets of vertices in a given graph G.

Lemma 4.1. *The collection $\{S_1, S_2, \ldots, S_k\}$ which is constructed by the algorithm RDS, forms a 2-separated collection of subsets of vertices in a graph G.*

Proof. Recall that a 2-separated collection is characterized by property A, i.e., the distance between any two vertices of two different subsets of the collection is more than 2. To prove the theorem, we use mathematical induction on z, the number of subsets in the 2-separated collection. The base step at $z = 1$, clearly, $\{S_1, V_2\}$ is a 2-separated collection in a graph G, since $V_2 = V/N[N[S_1]]$. Suppose by induction hypothesis at $z \leq i - 1$, that $\{S_1, S_2, \ldots, S_{i-1}, V_i\}$ is a 2-separated collection in a graph G. The distance from any vertex in V_i to another vertex in $S_1, S_2, \ldots, S_{i-1}$ is more than 2. Consider $V_{i+1} = V_i/N[N[S_i]]$, so V_{i+1} and S_i satisfy property A. Therefore, $\{S_1, S_2, \ldots, S_i, V_{i+1}\}$ is a 2-separated collection. ■

According to Lemma 2.2 and the fact that $D = \cup_{i=1}^{k} D(N_{\hat{r}+2}[v_i]) = \cup_{i=1}^{k} D(T_i)$ which forms a dominating set for the input graph G, we obtain the following Lemma:

Lemma 4.2. *The dominating set, D, which is computed by the algorithm RDS, is a(1.25)-approximation of a MDS in an arbitrary 5-degree graph.*

Proof. The two collections $\{S_1, S_2, \ldots, S_k\}$ and $\{T_1, T_2, \ldots, T_k\}$ which are constructed by the algorithm *RDS*, satisfy the following inequality:

$$|D(T_i)| \leq (1.25) \cdot |D(S_i)|, \quad \text{for all } i = 1, 2, \ldots, k.$$

Moreover, $\mathbb{D} = \cup_{i=1}^{k} D(T_i)$. Hence, we have

$$|\mathbb{D}| = \left| \bigcup_{i=1}^{k} D(T_i) \right| \leq \sum_{i=1}^{k} |D(T_i)| \leq (1.25) \cdot \sum_{i=1}^{k} |D(S_i)|.$$

By using Lemma 2.1, one can conclude that $(1.25) \cdot \sum_{i=1}^{k} |D(S_i)| \leq (1.25) \cdot |D(V)|$.

This completes the proof. ■

Finally, we claim that the running time of the algorithm is a polynomial. Evidently, the number of iterations of the outer-while loop (steps 3–16) is bounded by $n = |V|$. Thus, it is sufficient to compute the running time for only one iteration of this loop of the algorithm *RDS*. Note that the cardinality of $D(N_{r+2}[v])$ and $D(N_r[v])$ are bounded by $((r+2) \cdot \Delta^{\frac{1}{2}} + 1)$ and $(r \cdot \Delta^{\frac{1}{2}} + 1)$, respectively, where $\Delta = 5$. If r is a fixed integer (not large), then the minimum dominating sets $D(N_{r+2}[v])$ and $D(N_r[v])$ can be computed in a polynomial time. Therefore, the running time for one iteration is $O(n^h)$, where $h = O(r)$.

Hence, the algorithm *RDS* has polynomial running time. The next lemma shows that there exists a bound on $\hat{r}$, the first value of r which violates (4.1). This bound depends only on the approximation ($\rho = 1.25$), Δ, and not on the size of the input graph.

Table 1

No. of vertices	44	45	46	47	48
No. of chosen graphs randomly	3	3	3	3	3
Average of total time (minutes)	19. 536	31. 125	54. 321	149. 325	202. 118

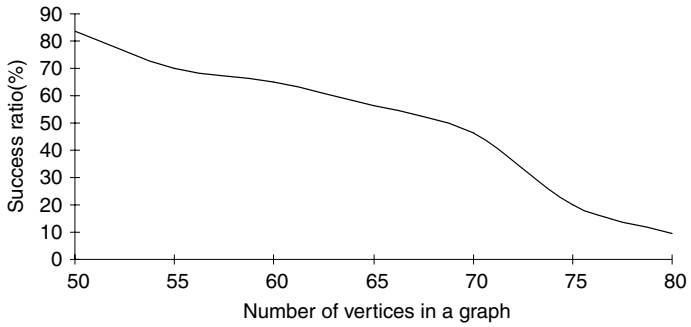

Fig. 1. The relation between the graph size and the success ratio.

Lemma 4.3. *There exists a function $c = c(\rho, \Delta)$ such that $\hat{r}$ is bounded by c, where $\hat{r}$ at which is the largest neighborhood, is constructed during an iteration of the outer-while loop (steps 3–16) of the algorithm RDS.*

Proof. It is clear that $|D(N_1[v])| = |D(N_0[v])| = 1$, because that the central vertex v dominates itself and all its neighbors. Consider an arbitrary value of $r < \hat{r}$.

The first case, if r is an even number, we have

$$((r+2)\cdot \Delta^{\frac{1}{2}} + 1) \geq |D(N_{r+2}[v])| > \rho \cdot |D(N_r[v])| > \cdots > \rho^{\frac{r+2}{2}} \cdot \quad |D(N_0[v])| = (\sqrt{\rho})^{r+2}. \tag{4.2}$$

The second case, if r is odd, we get

$$((r+2)\cdot \Delta^{\frac{1}{2}} + 1) \geq |D(N_{r+2}[v])| > \rho \cdot |D(N_r[v])| > \cdots > \rho^{\frac{r+1}{2}} \cdot |D(N_1[v])| = (\sqrt{\rho})^{r+1}. \tag{4.3}$$

Since $\rho > 1$, and so $\sqrt{\rho} > 1$, the above inequalities (4.2) and (4.3) have to be violated in both cases. The bound on r when these inequalities are violated for the first time, depends only on ρ, Δ, and not on the size of the input graph.

Let $f_i(r, \rho) = (\sqrt{\rho})^{r+i}$, $i = 1, 2$, and $g(r, \Delta) = ((r+2)\cdot \Delta^{\frac{1}{2}} + 1)$. It is clear that

$$\lim_{r\to\infty} \frac{f_i(r,\rho)}{g(r,\Delta)} \to \infty \quad \text{and} \quad \lim_{r\to\infty} \frac{g(r,\Delta)}{f_i(r,\rho)} \to 0.$$

This means that at a special value of r the following inequality is satisfied:

$$f_i(r, \rho) > g(r, \Delta), \quad i = 1, 2. \quad \blacksquare$$

Setting $c = \rho \cdot h \cdot \ln(\Delta)/\ln(\rho)$, $h = 1/(0.25)^2$ yields $((c+2)\cdot \Delta^{\frac{1}{2}} + 1) < (\sqrt{\rho})^c < f_i(c, \rho), i = 1, 2$. Thus, c is bounded by $O(\rho \cdot \ln(\Delta)/\ln(\rho))$.

The results of the *RDS*-algorithm

The algorithm *RDS* randomly chooses a 5-degree graph with fixed n vertices. Then, the algorithm can deal with all vertices of the chosen graph. We successfully executed the algorithm on arbitrary 5-degree graphs having n vertices for $n \leq 48$. During one complete run of the algorithm, if the vertices do not satisfy the stated condition in step 7 of the *RDS*-algorithm, then the approximation of a dominating set of the input graph is obtained. Otherwise, the algorithm terminates without any solution. We calculate the running time for getting the dominating set of each vertex. The algorithm is applied for some graphs chosen randomly.

To calculate the average running time for some graphs having the same number of vertices, first, calculate the average running time for every instance of them. This is done by executing the algorithm for every vertex in any instance and summing the running time for all vertices. Then, divide the running time of an instance by the number of its vertices. We do this for all instances to obtain the total average times, *TA*, of all underlying graphs having the same number of vertices. Lastly, the required average time is computed by dividing *TA* over the number of graphs having the same number of vertices. Table 1 includes the results of some studying cases.

From these results, the algorithm is not suitable for execution on all the vertices of graphs having more than 48 vertices since it has not taken a reasonable time. The algorithm is applied on the first ten vertices for some chosen graphs having more than 48 vertices. When the running time increased over two minutes, we did not complete the execution of the algorithm.

From Fig. 1 that illustrates the algorithm's success rate by the graph size, it is clear that the algorithm works for graphs with 50 vertices better than other cases. It is worthy to remark that increasing the size of the input graphs leads to an essential decrease of the algorithm's success rate.

When we executed the algorithm on graphs having 80 vertices, most groups of ten vertices had a running time of more than two minutes. For the testing instances in the above table, a very few number of vertices satisfy the condition in step 7 of the algorithm *RDS*. For example, only one graph from three testing graphs having 44 vertices has exactly one vertex, which satisfies this condition. The algorithm is implemented by the Java language. The program worked and recorded all results on a 2.02 GHz Pentium IV personal computer.

5. Concluding remarks

In this paper, a new randomized approximation algorithm for treating the problem of finding a *MDS* in a 5-degree graph is introduced. The algorithm succeeds in computing the (1.25) approximate dominating set in polynomial-time. Fortunately, in 5-degree graphs, the largest neighborhood which has to be considered in the algorithm is bounded by a constant that only depends on the desired approximation factor ($\rho = 1.25$) and not on the size of the input graph. The overall time complexity of the algorithm is $O(n^c)$, where $c = O(\rho \cdot \ln(\Delta)/\ln(\rho))$. This algorithm can be used for the minimum dominating set problem in d-degree graphs, where $d \geq 3$. When d is increased, the running time of the algorithm *RDS* is rapidly increased. Observe that this algorithm has a nice property which is: *The probability of getting a dominating set of the given graph can be increased by changing in lower bounds of* $|D(N_r[v])|$ *and* $|D(N_{r+2}[v])|$, *which are given in step 7 of the algorithm RDS.*

References

[1] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H Freeman & Co, New York, 1979.
[2] J. Hromkovič, Algorithmics for Hard Problems, Springer-Verlage, Berlin, 2001.
[3] T.W. Haynes, S.T. Hedetniemi, P.J. Slater, Fundamentals of Domination in Graphs, Marcel Dekker, Inc, New York, 1998.
[4] F.V. Fomin, D. Kratsch, G.J. Woeginger, Exact (exponential) algorithms for the dominating set problem, in: J. Hromkovič, M. Nagl, B. Westfechtel (Eds.), Proceedings of the 30th International Workshop on Graph-Theoretical Concepts in Computer Science WG'04, in: LNCS, vol. 3353, Springer-Verlage, 2004, pp. 245–256.
[5] T. Nieberg, J.L. Hurink, *A* PTAS for the minimum dominating set problem in unit disk graphs, University of Twente, Enschede, Memorandum 1732, 2004.
[6] F. Grandoni, A note on the complexity of minimum dominating set, J. Discrete Algorithms 4 (2006) 209–214.
[7] B. Gfeller, E. Vicari, A faster distributed approximation scheme for the connected dominating set problem for Growth-Bounded Graphs, in: Proceedings of the 6th International Conference on Ad – Hoc, Mobile, and Wireless Networks, AdHoc NOW, in: LNCS, vol. 4686, Springer-Verlage, Berlin, 2007, pp. 59–73.
[8] S. Butenko, X. Cheng, C. Oliveira, P.M. Pardalos, A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks, Cooperative Control and Optimization (2004) 61–73.