國 立 交 通 大 學

應用數學系

碩 士 論 文

一個用於無線網路中找出最小權重獨立控制集的多項式時間近似方案

A Polynomial-time Approximation Scheme for the Minimum-weighted Independent Dominating Set Problem in Wireless Networks

研 究 生：吳思賢

指導教授：陳秋媛　教授

中 華 民 國 九 十 九 年 六 月

一個用於無線網路中找出最小權重獨立控制集的多項式時間近似方案

A Polynomial-time Approximation Scheme for the Minimum-weighted Independent Dominating Set Problem in Wireless Networks

研 究 生：吳思賢　　Student：Sih-Sian Wu

指導教授：陳秋媛　　Advisor：Chiuyuan Chen

國 立 交 通 大 學

應 用 數 學 系

碩 士 論 文

A Thesis
Submitted to Department of Applied Mathematics
College of Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master
in
Applied Mathematics
June 2010
Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 九 年 六 月

一個用於無線網路中找出最小權重獨立控制集的

多項式時間近似方案

研究生：吳思賢　　　　　　　　　　　　　指導老師：陳秋媛　教授

國 立 交 通 大 學

應 用 數 學 系

摘　要

在無線網路中，每一個節點的能力不一定相同，例如，電池的剩餘電量不一定相同，訊息傳遞所需的花費不一定相同，因此，一個常見的方式是利用點權重圖(也就是將無線網路所對應的圖的點加上權重)來描述網路的拓樸結構。一個點權重圖的最小權重獨立控制集，指的是一組既是獨立集也是控制集的最小權重點集合。在本論文中，我們針對具有多項式成長有界性質的點權重圖提出一個計算最小權重獨立控制集的多項式時間近似方案；具有多項式成長有界性質的圖包含了著名的單位圓盤圖、單位圓球圖，和類單位圓盤圖。就我們所知，我們的多項式時間近似方案是第一個針對無線網路中最小權重獨立控制集問題的多項式時間近似方案。此外，當所有點的權重都一樣時，我們的多項式時間近似方案會是一個在具有多項式成長有界性質的圖中計算獨立控制集的多項式時間近似方案；值得一提的是：我們的多項式時間近似方案只需要一個階段，因此簡化了Hurink和Nieberg在文獻[15]中所提出需要兩個階段的多項式時間近似方案。

關鍵詞：控制集，獨立控制集，最小權重獨立控制集，近似演算法，多項式時間近似方案。

中 華 民 國 九 十 九 年 六 月

A polynomial-time approximation scheme for the minimum-weighted independent dominating set problem in wireless networks

Student: Sih-Sian Wu Advisor: Chiuyuan Chen

Department of Applied Mathematics
National Chiao Tung University

Abstract

Due to different capabilities of nodes (for example, different remaining battery lives or different costs for transmitting a message) in a wireless network, it is desirable to model the underlying network topology by a node-weighted graph. A minimum-weighted independent dominating set of a node-weighted graph is a vertex subset with the minimum weight being both independent and dominating. In this thesis, we present a polynomial-time approximation scheme (PTAS) for computing a minimum-weighted independent dominating set in a node-weighted polynomially growth bounded graph, which is in a class of graphs that include the well-known unit disk graphs, unit ball graphs, and quasi unit disk graphs. To the best of our knowledge, our PTAS is the first PTAS for the minimum-weighted independent dominating set problem in wireless networks. Furthermore, when all the weights are identical, our PTAS turns out to be a simple 1-stage PTAS for computing an independent dominating set in a polynomially growth bounded graph and hence simplifies the 2-stage PTAS proposed by Hurink and Nieberg in [15].

Key words: Dominating set; Independent dominating set; Minimum-weighted independent dominating set; Approximation algorithms; Polynomial-time approximation scheme.

誌謝

能完成這篇畢業論文，最需要感謝的就是我的指導教授－陳秋媛教授。在我遇到瓶頸的時候，老師總是有耐心地教導我，給我一個正確的方向，這讓我的研究能順利地完成。老師對我的照顧不僅止於課業上，老師也時常關心我的生活，讓我能無憂無慮的進行研究。

再來我要感謝的是國元學長，學長幫助我完成了整篇論文的架構，還有給我數據模擬的方向。對於我的問題，學長也總是不厭其煩地回答我。我真的非常感謝學長這兩年來的指導。

要感謝的人很多，不管是最會解決問題的鈺傑學長，還是時常跟我一起討論問題的思綸、健峰，還有在我做研究苦悶時，一起打球的同學，真的很謝謝大家在我研究所這段期間的幫忙。

Contents

List of Figures

1. Introduction

1.1. Minimum independent dominating set problem

Most works on optimization problems that arise in wireless networks model networks by graphs. In this thesis, graphs refer to simple undirected graphs. Let $G = (V, E)$ be a graph with vertex set V and edge set E. A subset I of V is an *independent set* if no two vertices in I are adjacent. A subset D of V is a *dominating set* if every vertex in $V \setminus D$ has a neighbor in D. The following two problems have been extensively studied in the literature: the *Maximum Independent Set Problem* (**MIS**) and the *Minimum Dominating Set Problem* (**MDS**). The former asks for finding an independent set with the maximum cardinality and the latter, finding a dominating set with the minimum cardinality. Since the above two problems have been extensively studied in the literature [5, 6, 7, 9, 11, 12, 13, 16, 17, 18], the *Minimum Independent Dominating Set Problem* (**MIDS**), which is to find a subset of vertices that is both an independent and a dominating set with the smallest cardinality, has received much attention in the literature [2, 9, 15, 19]. A *maximal independent set* is an independent set that is not a subset of any other independent set. It is well-known that any independent dominating set is a maximal independent set, and vice versa. Therefore, **MIDS** is also known as the *Minimum Maximal Independent Set Problem.*

It has been proven that **MIDS** is NP-complete in general graphs [12]. Furthermore, Halldórsson [13] showed that **MIDS** in general graphs can not be approximated within the ratio $n^{1-\varepsilon}$ unless $P = NP$. In addition, **MIDS** remains NP-complete under strict restrictions: for example, for bipartite graphs and comparability graphs [7], for planar graphs, triangle-free graphs and claw-free graphs [2], for line graphs [20], and for $2P_3$-free graphs [19]. On the other hand, **MIDS** can be solved in polynomial time for chordal graphs [9], cocomparability graphs [16], and asteroidal triple-free graphs [3]. In [15], Hurink and Nieberg presented a 2-stage polynomial-time approximation scheme (PTAS) for **MIDS** in polynomially growth bounded graphs. For more details, see [2, 19].

1.2. Minimum-weighted independent dominating set problem

In real world applications, nodes in a wireless network usually have different capabilities (for example, different remaining battery lives or different costs for transmitting a message). Therefore, in this thesis we assume there are positive weights on vertices of a graph and we call such a graph a *node-weighted graph*. For convenience, a vertex of a graph is also called a node. The *Minimum-Weighted Independent Dominating Set Problem* (**MWIDS**) is to find an independent dominating set Ω of a node-weighted graph such that the sum of weights of nodes in Ω is minimized. **MWIDS** is clearly a generalization of **MIDS** since the latter can be reduced to the former by letting all nodes have the same positive weight.

In this thesis, we mainly concern with **MWIDS** in node-weighted polynomially growth bounded graphs. The class of polynomially growth bounded graphs includes most of the graphs used to model wireless networks such as Unit Disk Graphs, Quasi Unit Disk Graphs, Unit Ball Graphs, and Coverage Area Graphs [15, 17], where a Unit Disk Graph (UDG) is defined as the intersection graph of equal radius disks in the Euclidean plane (see Figure 1). Since **MIDS** is a special case of **MWIDS** and **MIDS** is NP-complete in general graphs, it follows that **MWIDS** is NP-complete in general graphs. Chang [4] showed that **MWIDS** is NP-complete for chordal graphs. Farber presented a linear time algorithm to locate a minimum weight independent dominating set in a chordal graph with 0-1 vertex weights [9], which is a bit different from the original **MWIDS** problem. Farber et al. also used polynomial time to deal with **MWIDS** in permutation graphs [11] and strongly chordal graphs [10].

For node-weighted polynomially growth bounded graphs, many researchers considered the *Minimum-Weighted Dominating Set Problem* (**MWDS**) and the *Minimum-Weighted Connected Dominating Set Problem* (**MWCDS**); see [1, 8, 14, 21, 22]. Till now, the best

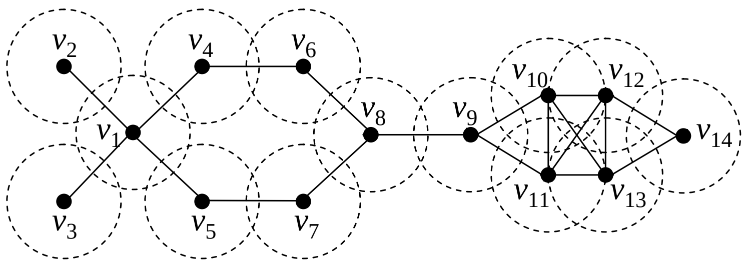


Figure 1: A unit disk graph.

known approximation ratios for **MWDS** and **MWCDS** in UDGs are obtined by Zou et al. in [22]; they presented a $(4+\varepsilon)$-approximation algorithm for **MWDS** and a $(5+\varepsilon)$-approximation algorithm for **MWCDS** in UDGs. To the best of our knowledge, there is no related result for **MWIDS** in polynomially growth bounded graphs in the literature.

1.3. Our result

In this thesis, we present a PTAS for **MWIDS** in a subclass of node-weighted polynomially growth bounded graphs. In particular, we prove that if the weight function of a node-weighted graph satisfies the following weight-constraint, then there exists a PTAS for **MWIDS**.

Weight-constraint: There exists a constant $t > 0$ such that for every pair u, v of nodes of a node-weighted graph G, $w(u) \leq t \cdot w(v)$ holds.
This constraint is reasonable in real world applications as, for example, the difference between the remaining battery lives of any two nodes or the difference between the costs for transmitting a message from any two nodes cannot be unbounded. Do notice that by assigning the same weight to every node, our PTAS turns out to be a simple 1-stage PTAS for **MIDS** in polynomially growth bounded graphs and hence simplifies the 2-stage PTAS proposed by Hurink and Nieberg in [15]. For convenience, we call the PTAS proposed by Hurink and Nieberg in [15] HN PTAS in the remaining sections.

This thesis is organized as follows: Section 2 gives the preliminaries including the definitions and terminologies. We present our PTAS for **MWIDS** in polynomially growth bounded graphs in Section 3. A comparison between Our PTAS with HN PTAS are given in Section 4. Concluding remarks are given in Section 5.

2. Preliminaries

polynomially growth bounded

2.1. polynomially growth bounded graphs

For any two vertices u and v of G, let $d(u, v)$ denote the number of hops in a shortest path from u to v. The following definitions are crucial in this thesis. The *r-neighborhood* of v, denoted by $N_r[v]$, is defined by

$$N_r[v] = \{u \in V \mid d(u, v) \leq r\}.$$

So, $N_0[v] = \{v\}$ and $N_1[v]$ is the set of v and the neighbors of v. For convenience, let $N[v] = N_1[v]$. Analogously, for a subset U of V, let $N[U] = \bigcup_{u \in U} N[u]$. The following definition is given by Hurink and Nieberg [15].

Definition 1. Let $G = (V, E)$ be a graph. If there exists a function $f(\cdot)$ such that every *r-neighborhood* in G contains at most $f(r)$ independent vertices, then G is *f-growth-bounded.* Furthermore, we say that G has *polynomially* bouuded growth is for some constant $c \geq 1$, $f(r)$ is bounded by a polynomial of maximal degree c, i.e., $f(r) = O(r^c)$.

In [15], Hurink and Nieberg mentioned that the growth function $f(\cdot)$ does not depend on the number of vertices in the given graph, but on the radius of the neighborhoods only. For example, a UDG has $p(r) = (2r + 1)^2 = O(r^2)$; see [18].

2.2. Terminologies

For a given optimization problem, an algorithm is called a *ρ-approximation algorithm* for some $\rho \geq 1$ if it always returns a feasible solution of relative error no more than ρ. Here ρ is called the *approximation ratio* and the solution derived is called a *ρ-approximation.*

For a given optimization problem, an algorithm is called a *polynomial-time approximation scheme* (PTAS) if it is an approximation algorithm that takes as input not only an instance of the problem but also a value $\varepsilon > 0$ such that it is a $(1+\varepsilon)$-approximation algorithm and for any fixed $\varepsilon > 0$, it runs in time polynomial in the size n of its input instance.

In the remaining discussion, we assume that $G = (V, E)$ is a node-weighted p-growth-bounded graph where $p(\cdot)$ is a polynomial function, and n is used to denote the number of vertices of G. For a subset U of V, the subgraph induced by U is denoted by $G[U]$. Let W_{opt} be a minimum-weighted independent dominating set of G and w_{opt} be the weight of W_{opt}, where the weight of a vertex subset is the sum of weights of all vertices in this subset. For a subset of vertices U of G, define $W^*(U)$ to be a function that returns a minimum-weighted independent dominating set for U in G. We denote $w^*(U)$ to be the weight of $W^*(U)$. Note that $W^*(U)$ is always computed with respect to the entire graph G which is updated after an iteration of algorithm; therefore, the set returned by $W^*(U)$ may contain vertices outside U (but the set returned by $W^*(U)$ is always contained in $N[U]$). For example, consider in Figure 2. Suppose all nodes have the same weight. Then $W^*(\{v_1, v_2\}) = \{v_3\} \not\subseteq \{v_1, v_2\}$, but $W^*(\{v_1, v_2\}) \subseteq N[\{v_1, v_2\}]$. Clearly,

$$W_{opt} = W^*(V) \text{ and } w_{opt} = w^*(V).$$

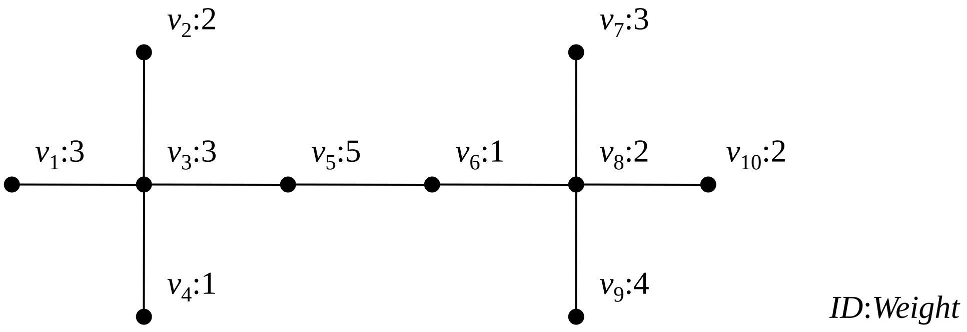


Figure 2: An example of node-weighted graph with $W_{opt} = \{v_3, v_8\}$ and $w_{opt} = 5$.

A *d-separated collection* is a collection $\{S_1, S_2, \ldots, S_k\}$, where $S_i \subseteq V,\ i = 1, 2, \ldots, k$, with the property that for any two vertices $s \in S_i,\ \overline{s} \in S_j,\ i \neq j,\ d(s, \overline{s}) > d$ holds. The subgraphs induced by subsets in a d-separated collection divide the original graph into smaller parts so that it becomes easier to solve a given optimization problem. In [18], Nieberg and Hurink successfully used a 2-separated collection to derive a PTAS for **MDS** in polynomially growth bounded graphs. They also used the same idea to derive a PTAS for **MIDS** in polynomially growth bounded graphs in [15].

3. A PTAS for the MWIDS problem on node-weighted polynomially growth bounded graphs

In order to present our PTAS, we first propose statement (1) and Lemma 3.1. For $v \in V$, let $I_r[v]$ denote a maximum independent set of $N_r[v]$. Since a maximum independent set is also a dominating set, we have

$$| W^*(N_r[v]) | \leq | I_r[v] | \leq p(r). \tag{1}$$

The following lemma says that obtaining an optimal solution $W^*(N_r[v])$ for $N_r[v]$ can be done in polynomial time if r is bounded.

Lemma 3.1. *Let $G = (V, E)$ be a node-weighted graph. Then for any neighborhood $N_r[v]$, we can obtain $W^*(N_r[v])$ in $n^{O(p(r))}$ time.*

Proof. By statement 1, we know the cardinality of $W^*(N_r[v])$ is at most p(r). It becomes clear that we can obtain $W^*(N_r[v])$ in $n^{O(p(r))}$ time. $\square$

Before going further, we need two lemmas.

Lemma 3.2. *Let $G = (V, E)$ be a node-weighted graph. For a d-separated collection $\{S_1, S_2, \ldots, S_k\}$, where $S_i \subseteq V$, $i = 1, 2, \ldots, k$, in G with $d \geq 2$, we have*

$$w_{opt} \geq \textstyle\sum_{i=1}^{k} w^*(S_i).$$

Proof. For each subset $S_i \subseteq V$, consider the neighborhood $N[S_i]$. By definition, $N[S_i]$ and $N[S_j]$ are pairwise disjoint, for all $i, j \in \{1, 2, \ldots, k\}$, $i \neq j$, and thus we get $w_{opt} \geq$

$\sum_{i=1}^{k} w(W_{opt} \cap N[S_i])$. Since $W_{opt} \cap N[S_i]$ dominates S_i and is independent in G, we have $w(W_{opt} \cap N[S_i]) \geq w^*(S_i)$. This implies

$$w_{opt} \geq \sum_{i=1}^{k} w(W_{opt} \cap N[S_i]) \geq \sum_{i=1}^{k} w^*(S_i).$$

□

Lemma 3.3. *Let $\{S_1, S_2, \ldots, S_k\}$ be a d-separated collection in a node-weighted graph $G = (V, E)$, $d \geq 2$, and let $T_1, T_2, \ldots, T_k$ be subsets of V with $S_i \subseteq T_i$ for all $i = 1, 2, \ldots, k$. If there exists a bound $\rho \geq 1$ such that $w^*(T_i) \leq \rho \cdot w^*(S_i)$ holds for all $i = 1, 2, \ldots, k$, then*

$$\sum_{i=1}^{k} w^*(T_i) \leq \rho \cdot w_{opt}.$$

Moreover, if $\bigcup_{i=1}^{k} W^*(T_i)$ is a minimum-weighted independent dominating set of G, then it is a ρ-approximation of a minimum-weighted independent dominating set of G.

Proof. This lemma follows from

$$\sum_{i=1}^{k} w^*(T_i) \leq \sum_{i=1}^{k} \rho \cdot w^*(S_i) \leq \rho \cdot w_{opt}.$$

□

We now are ready to introduce our PTAS; it is presented in Algorithm 1.

Our PTAS works as follows. Initially, set the independent dominating set Ω to be the empty set. Start with an arbitrary vertex $v \in V$ and consider the r-neighborhood $N_r[v]$, for $r = 0, 1, 2$, and so on. For each r, compute minimum-weighted independent dominating sets for the neighborhoods $N_{r+2}[v]$ and $N_r[v]$ and compute their total weight whenever

$$w^*(N_{r+2}[v]) > (1 + \varepsilon) \cdot w^*(N_r[v]).$$

After stopping increasing the radius r of the neighborhoods, the desired bound on the weight of an independent dominating set is obtained. Update Ω and update the remaining vertices. Repeat the above process until V is empty.

Algorithm 1 An algorithm for the **MWIDS** problem.

Input: a node-weighted polynomially growth bounded graph $G = (V, E)$ and a constant $\varepsilon > 0$

Output: an independent dominating set Ω of G which is a $(1+\varepsilon)$-approximation for the **MWIDS** problem

```
1:  $\Omega = \varnothing$; $i = 0$;
2:  while $V \neq \varnothing$ do
3:      $i = i + 1$;                                         ▷ the $i$-th iteration
4:      Pick $v \in V$;
5:      $r = 0$;
6:      while $w^*(N_{r+2}[v]) > (1+\varepsilon) \cdot w^*(N_r[v])$ do
7:          $r = r + 1$;
8:      end while
9:      $\Omega = \Omega \cup W^*(N_{r+2}[v])$;
10:     $V = V \setminus N[W^*(N_{r+2}[v])]$;
11: end while
12: return $\Omega$;
```

We now analyze Algorithm 1. First of all, the following lemma shows that the radius of the largest neighborhood we need to examine for each vertex picked in line 3 of Algorithm 1 is bounded by a constant that only depends on the growth function p and the given constant ε. In other words, the number of iterations of the inner while-loop is constant.

Lemma 3.4. *Let $G = (V, E)$ be a node-weighted p-growth-bounded graph and the weight function satisfies the weight-constraint. Then for the i-th iteration of Algorithm 1, there exists a constant c (only depends on ε) such that the number of iterations in lines 6-8 is at most c.*

Proof. Let the growth function p be $p(r) = O(r^d)$ for some constant $d \geq 1$. Consider a vertex v picked in line 3 in the i-th iteration of Algorithm 1. Let $\hat{r}$ denote the first r which violates the inequality in line 6. Now we consider any value $r < \hat{r}$. Let v^* be the vertex in $N_{r+2}[v]$ with the maximum weight. If r is even, then by (1), we have

$$\begin{aligned}
p(r+2) \cdot w(v^*) &\geq w^*(N_{r+2}[v]) \\
&> (1+\varepsilon) \cdot w^*(N_r[v]) \\
&\vdots \\
&> (1+\varepsilon)^{\frac{r}{2}+1} \cdot w^*(N_0[v]) \\
&= (1+\varepsilon)^{\frac{r}{2}+1} \cdot w(x), \text{ for some } x \in N[v].
\end{aligned}$$

Recall that there exists a constant $t > 0$ such that for every pair of vertices u, v in G, $w(u) \leq t \cdot w(v)$ holds. Hence we have $(1+\varepsilon)^{\frac{r}{2}+1} < t \cdot p(r+2) = O((r+2)^d)$; this inequality will be violated for some large enough r. Therefore, there must exist a constant c (which depends on ε) such that $r \leq c$ holds. Similar arguments can be applied for odd r. $\square$

Theorem 3.5. *Algorithm 1 is a PTAS with approximation ratio $(1+\varepsilon)$ for the **MWIDS** problem on node-weighted polynomially growth bounded graphs $G = (V, E)$ if the weight function satisfies the weight-constraint.*

Proof. We first prove that Algorithm 1 generates a minimum-weighted independent dominating set with approximation ratio $(1+\varepsilon)$. Suppose at the end of Algorithm 1, the value of i is k; that is, suppose there are total k iterations of the while-loop in lines 2-11. For each i, let v_i be the vertex picked in line 3 and let r_i be the radius which is the first value violating the inequality in line 6. Then we have

$$w^*(N_{r_i+2}[v_i]) \leq (1+\varepsilon) \cdot w^*(N_{r_i}[v_i]). \tag{2}$$

Let $S_i = N_{r_i}[v_i]$ and $T_i = N_{r_i+2}[v_i]$ for $i = 1, 2, \ldots, k$. It is not difficult to verify that $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ forms a 2-separated collection. Clearly, $S_i \subseteq T_i$. By (2), $w^*(T_i) \leq (1+\varepsilon) \cdot w^*(S_i)$ holds for all $i = 1, 2, \ldots, k$. By Lemma 3.3, $\sum_{i=1}^{k} w^*(T_i) \leq (1+\varepsilon) \cdot w_{opt}$.

Algorithm 1 returns $\Omega = \bigcup_{i=1}^{k} W^*(T_i)$. If we can further prove that Ω is an independent dominating set of G, then by Lemma 3.3, Ω is a $(1+\varepsilon)$-approximation for a minimum-weighted independent dominating set of G. Ω is a dominating set of G since in the i-iteration, the removed neighborhood $N[W^*(N_{r_i+2}[v_i])]$ is dominated by $W^*(T_i)$. Ω is an independent set of G since for each i, $W^*(T_i)$ is independent and if there exist two vertices a, b with $a \in W^*(T_i), b \in W^*(T_j)$, for some $i \neq j$, then $b \notin N[W^*(T_i)]$ and $a \notin N[W^*(T_j)]$ and therefore $W^*(T_i) \cup W^*(T_j)$ is also independent. From the above discussion, Ω is a $(1+\varepsilon)$-approximation of a minimum-weighted independent dominating set of G.

By Lemma 3.4, the radius of the largest neighborhood we need to consider is bounded by a constant $c = c(\varepsilon)$. It has been shown in [15] and [18] that $c(\varepsilon) = O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$. By Lemma 3.1, for each i, finding each of the local optimal solutions $W^*(N_0[v_i])$, $W^*(N_1[v_i])$, $\ldots$, $W^*(N_{r_i+2}[v_i])$ can be done in polynomial time $n^{O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})}$. Thus the total execution time spent on line 6 is bounded by a polynomial in n. Since the overall running time of Algorithm 1 is dominated by the total execution time spent on line 6, Algorithm 1 takes polynomial time. □

4. Comparing our PTAS with HN PTAS

To compare our PTAS with HN PTAS, we vary the number of nodes n from 50 to 100 with the increment of 10 and for each n, we generate one hundred unit disk graphs (UDGs). The method to generate a UDG with n nodes is to randomly place n nodes in a square of area size $100 \times 100m^2$. If the distance between any two nodes is within 1, there is an edge between them. It is well known that UDGs are polynomially growth bounded graphs.

HN PTAS [15] is a 2-stage algorithm; see Algorithm 2 and Algorithm 3 in Appendix. If the output of the first stage is not an independent set, then the second stage will be initiated in order to deal with those dependent vertices. The second stage will remove non-independent vertices from the output of first stage and adds independent ones by using maximal independent set scheme on vertices which have not been dominated. We implement our PTAS and HN PTAS by using the C programming language. See Figures 3, 4 and 5 for an example. Note that the vertices colored red in Figures 4 and 5 are the vertices in the output (i.e., the independent dominating set).

From Figure 4(a), we know that the first stage of HN PTAS produces 12 vertices and two of them are dependent. Thus the second stage of HN PTAS is initiated to tackle the problem; one of the two dependent vertices is removed and two vertices are added to the output; see Figure 4(b). The cardinality of the output of HN PTAS is therefore 13. For

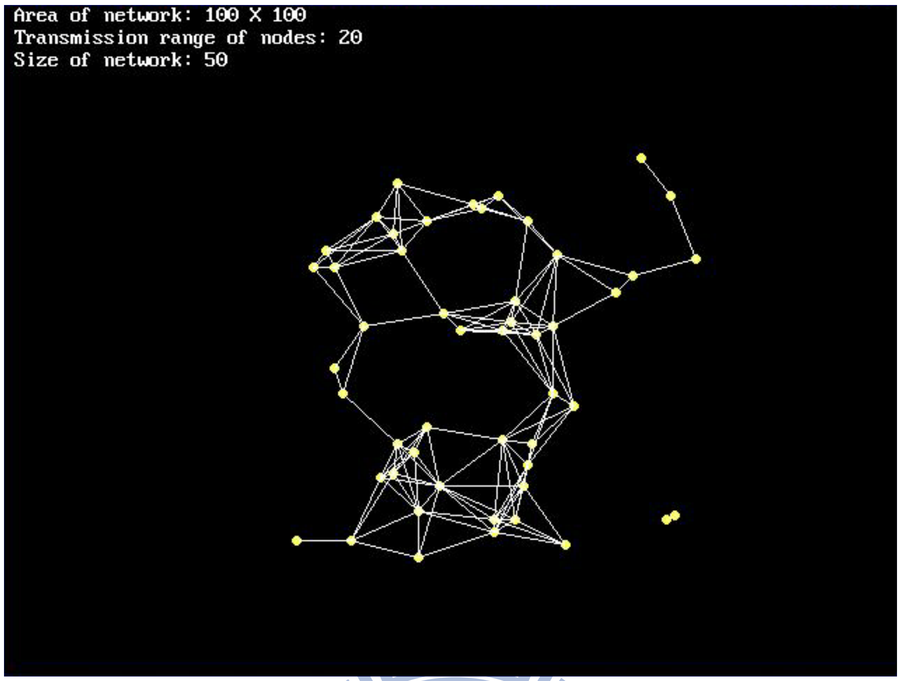
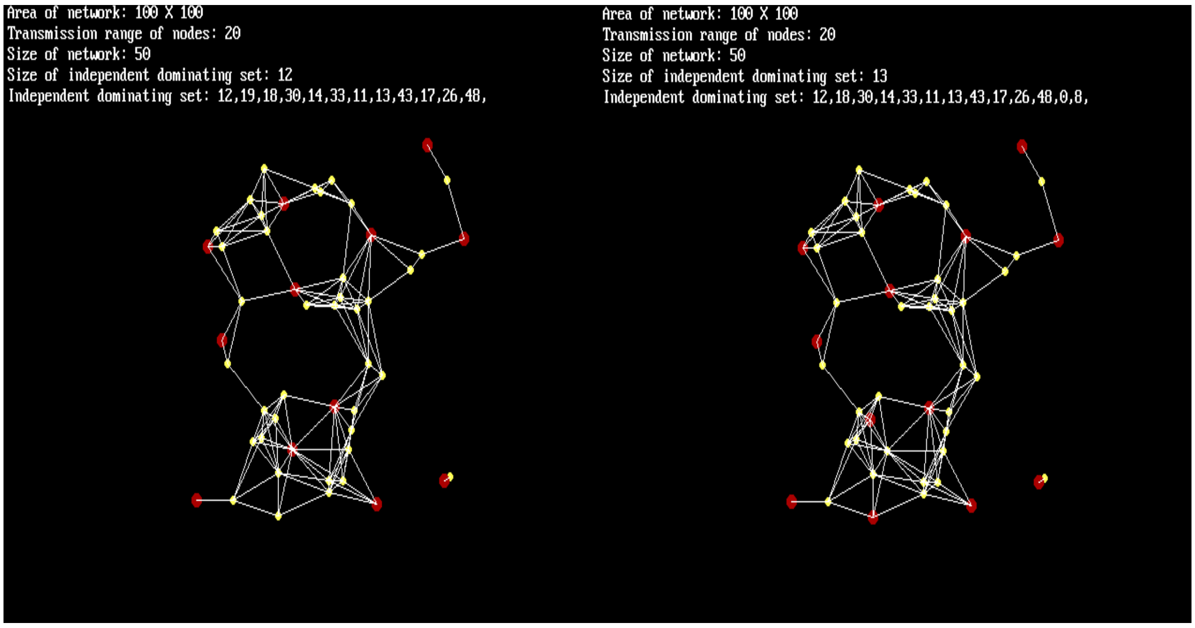


Figure 3: An input graph; $\varepsilon = 1.5$ and radius $= 20$.

the same graph, the cardinality of the output of our PTAS is only 10; see Figure 5.

From the experimental results, we observe that the second stage of HN PTAS may add many vertices to the output of the first stage. We observe that the performance of our PTAS (when considering the cardinality of the generated independent dominating set) outperforms that of HN PTAS; see Figures 6 and 7.

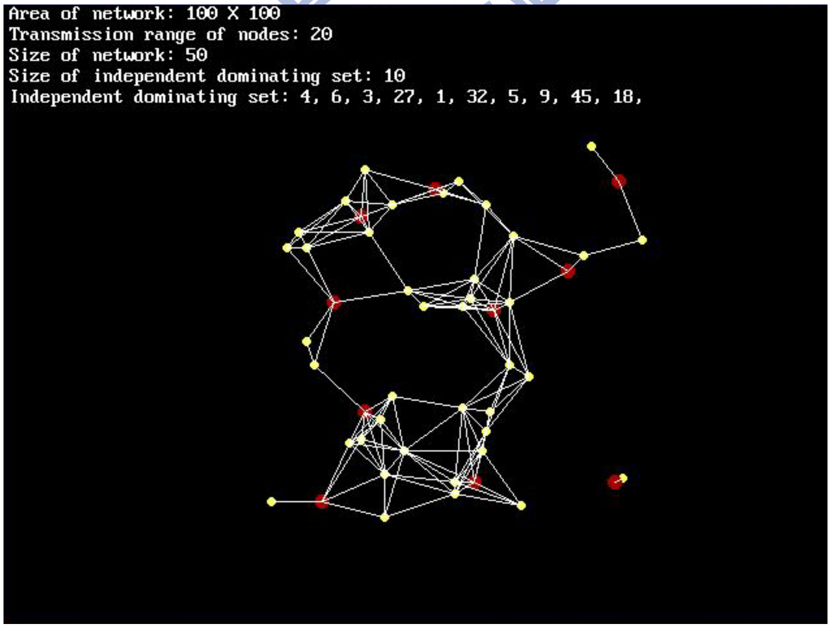
Note that both our PTAS and HN PTAS need to use brute force to obtain local optimal solutions. Since the local area that our PTAS needs to apply the brute force method is always no greater than that of HN PTAS, it is clear that our PTAS is more efficient than HN PTAS. But, how efficient? We give a threshold of 10 seconds, meaning that if a brute force execution exceeds 10 seconds, then the corresponding graph will be removed from the experimental set. From Figures 8 and 9, we see that the successful ratio (the ratio of graphs that do not exceed the threshold of 10 seconds) of our PTAS outperforms that of HN PTAS.



```
Area of network: 100 X 100
Transmission range of nodes: 20
Size of network: 50
Size of independent dominating set: 12
Independent dominating set: 12,19,18,30,14,33,11,13,43,17,26,48,
```

```
Area of network: 100 X 100
Transmission range of nodes: 20
Size of network: 50
Size of independent dominating set: 13
Independent dominating set: 12,18,30,14,33,11,13,43,17,26,48,0,8,
```

Figure 4: An output of HN PTAS when $\varepsilon = 1.5$ and radius $= 20$: (a) after stage 1, (b) after stage 2.



```
Area of network: 100 X 100
Transmission range of nodes: 20
Size of network: 50
Size of independent dominating set: 10
Independent dominating set: 4, 6, 3, 27, 1, 32, 5, 9, 45, 18,
```

Figure 5: An output of our PTAS when $\varepsilon = 1.5$ and radius $= 20$.

5. Concluding remarks

In this thesis, we consider the minimum-weighted independent dominating set problem on node-weighted polynomially growth bounded graphs. We have proven that if the weight function satisfies the weight-constraint: for every pair u, v of vertices, there exists a constant $t > 0$ such that $w(u) \leq t \cdot w(v)$ holds, then our algorithm is a PTAS for

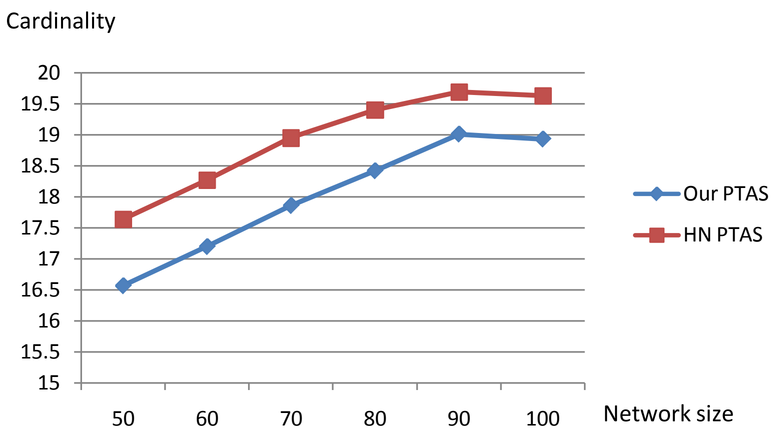


Figure 6: Comparing the cardinality of independent dominating set when $\varepsilon = 1.5$ and radius $= 15$.

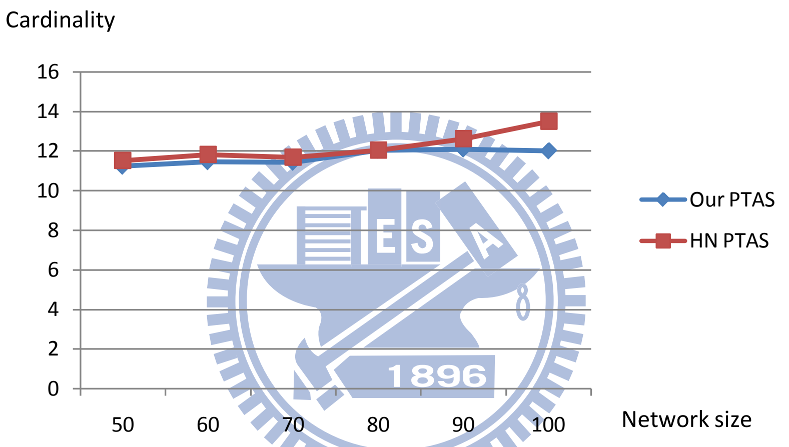


Figure 7: Comparing the cardinality of independent dominating set when $\varepsilon = 1.5$ and radius $= 20$.

the **MWIDS** problem on node-weighted polynomially growth bounded graphs. To the best of our knowledge, this is the first result for the MWIDS problem. Our PTAS has approximation ratio $(1+\varepsilon)$ and it can be done in polynomial time $n^{O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})}$. Our algorithm can also be used to solve the minimum independent dominating set problem.

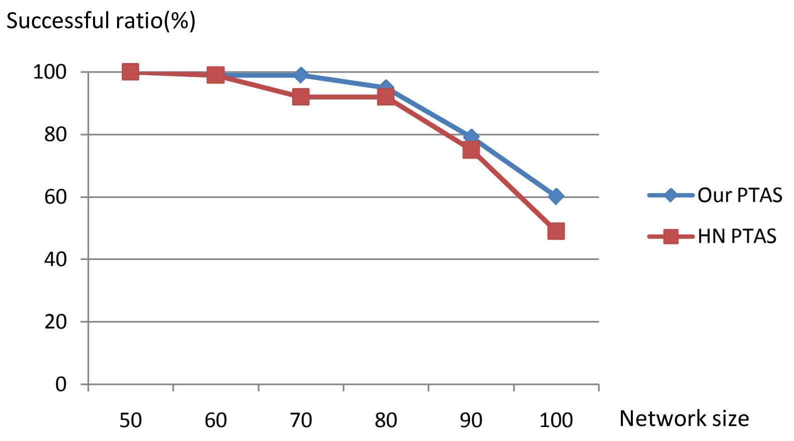


Figure 8: A comparison on successful ratio of algorithm in case ε =1.5, radius = 15.

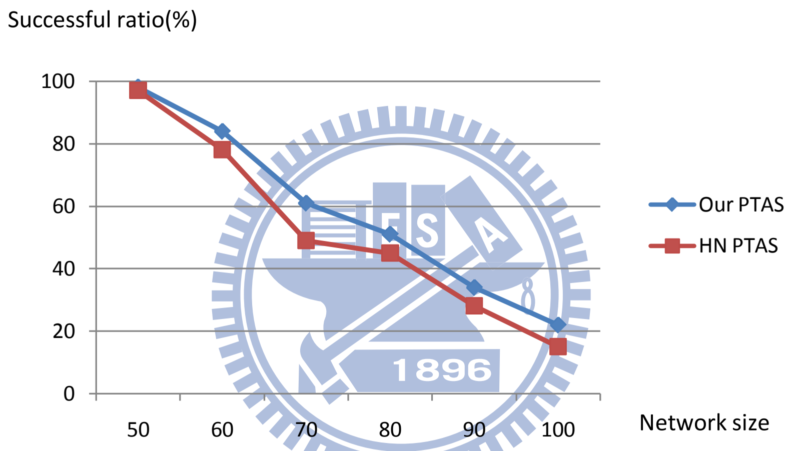


Figure 9: A comparison on successful ratio of algorithm in case ε =1.5, radius = 20.

References

[1] Ambuhl, C., Erlebach, T., Mihalak, M., Nunkesser, M., 2006. Constant-factor approximation for minimum-weight (connect) dominating sets in unit disk graphs. In: Lecture Notes in Computer Science 4110. pp. 3–14.

[2] Boliac, R., Lozin, V., 2003. Independent domination in finitely defined classes of graphs. Theoretical Computer Science 301 (1-3), 271–284.

[3] Broersma, H., Kloks, T., Kratsch, D., Müller, H., 1999. Independent sets in asteroidal triple-free graphs. SIAM Journal on Discrete Mathematics 12 (2), 276–287.

[4] Chang, G., 2004. The weighted independent domination problem is NP-complete for chordal graphs. Discrete Applied Mathematics 143 (1-3), 351–352.

[5] Chang, M., 1998. Efficient algorithms for the domination problems on interval and circular-arc graphs. SIAM Journal on Computing 27 (6), 1671–1694.

[6] Cheng, X., Huang, X., Li, D., Wu, W., Du, D., 2003. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. Networks 42 (4), 202–208.

[7] Corneil, D., Perl, Y., 1984. Clustering and domination in perfect graphs. Discrete Applied Mathematics 9 (1), 27–39.

[8] Dai, D., Yu, C., 2009. A $5 + \varepsilon$-approximation algorithm for minimum weighted dominating set in unit disk graph. Theoretical Computer Science 410 (8-10), 756–765.

[9] Farber, M., 1982. Independent domination in chordal graphs. Operations Research Letters 1 (4), 134–138.

[10] Farber, M., 1984. Domination, independent domination, and duality in strongly chordal graphs. Discrete Applied Mathematics 7 (2), 115–130.

[11] Farber, M., Keil, J. M., 1985. Domination in permutation graphs. Journal of Algorithms 6 (3), 309–321.

[12] Garey, M. R., Johnson, D. S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA.

[13] Halldórsson, M. M., 1993. Approximating the minimum maximal independence number. Information Processing Letters 46 (4), 169–172.

[14] Huang, Y., Gao, X., Zhang, Z., Wu, W., 2008. A better constant-factor approximation for weighted dominating set in unit disk graph. Journal of Combination Optimization 18 (2), 179–194.

[15] Hurink, J. L., Nieberg, T., 2008. Approximating minimum independent dominating sets in wireless networks. Information Processing Letters 109 (2), 155–160.

[16] Kratsch, D., Stewart, L., 1993. Domination on cocomparability graphs. SIAM Journal on Discrete Mathematics 6 (3), 400–417.

[17] Nieberg, T., Hurink, J., 2004. Wireless communication graphs. In: Palaniswami, M and Krishnamachari, B and Challa, S (Ed.), Proceedings of the 2004 Intelligent Sensors, Sensor Networks & Information Processing Conference. pp. 367–372, Intelligent Sensors, Sensor Networks and Information Processing Conference, Melbourne, Australia, Dec. 14-17, 2004.

[18] Nieberg, T., Hurink, J., 2006. A PTAS for the minimum dominating set problem in unit disk graphs. Vol. 3879 of Lecture Notes in Computer Science. pp. 296–306, 3rd International Workshop on Approxination and Online Algorithms, Palma de Mallorca, Spain, Oct. 06-07, 2005.

[19] Orlovich, Y. L., Gordon, V. S., de Werra, D., 2009. On the inapproximability of independent domination in $2P_3$-free perfect graphs. Theoretical Computer Science 410 (8-10), 977–982.

[20] Yannakakis, M., Gavril, F., 1980. Edge dominating sets in graphs. SIAM Journal on Applied Mathematics 38 (3), 364–372.

[21] Zou, F., Li, X., Gao, S., Wu, W., 2009. Node-weighted steiner tree approximation in unit disk graphs. Journal of Combination Optimization.

[22] Zou, F., Wang, Y., Xu, X.-H., Li, Du, H., Wan, P., Wu, W., 2009. New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. Theoretical Computer Science.

6. Appendix

Algorithm 2 The first stage of HN PTAS.

```
Input: $G = (V, E)$ polynomially growth bounded graph $\varepsilon > 0$
Output: Dominating set $\bar{D}$
 1: $\bar{D} = \varnothing$; $i = 0$;
 2: while $V \neq \varnothing$ do
 3:     Pick $v \in V$;
 4:     $r_i = 0$;
 5:     while $D^{(i)}_{r_i+3}(v) > (1 + \varepsilon) \cdot D^{(i)}_{r_i}(v)$ do
 6:         $r_i = r_i + 1$;
 7:     end while
 8:     Color vertices in $D^{(i)}_{r_i+3}(v)$ with color $i$;
 9:     $\bar{D} = \bar{D} \cup D^{(i)}_{r_i+3}(v)$;
10:     $V = V \setminus \Gamma_{r_i+3}(v)$;
11:     $i = i + 1$;
12: end while
```

Algorithm 3 The second stage of HN PTAS.

```
Input: $G = (V, E)$, Dominating set $\bar{D}$, $v \in \bar{D}$
1: Assert that $v$ is conflicting;
2: $\bar{D} = \bar{D} \setminus \{v\}$;
3: $V' = V \setminus \Gamma(\bar{D})$;
4: Compute maximal independent set $I$ on $G[V']$;
5: $\bar{D} = \bar{D} \cup I$;
```

```
//File Name: PTAS_for_MIDS.cpp
//Author:  吳思賢
//Email Address: smallhau@gmail.com
//Description: This program is a PTAS (polynomial-time approximation scheme)
//      for the minimum independent dominating set problem on unit disk graphs.
//Input: One hundred adjacency matrices; each matrix represents a unit disk
//      graph and the entries of each matrix are generated randomly.
//Output: The average cardinality and the average execution time for generating
//      the independent dominating sets of the one hundred unit disk graphs.

#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <time.h>
#include <cstdlib>
using namespace std;

#define NETWORK_SIZE 50        //設定 NETWORK 的 size.

int check(int adj[ ][NETWORK_SIZE+1], int IDS[ ], int N[ ], int nb_size, int size);
//目的：利用 adjacency matrix adj 來判斷所選的點集 IDS 是否真的是
//      an independent dominating set of a given r-Neighborhood N.
//需給傳入值的參數：adj, IDS, N, id, size
//需做傳出值的參數：無
//傳回：1 表示判斷之結果爲"是"；0 表示判斷之結果爲"否".
//參數 adj：存放 unit disk graph 的 adjacency matrix.
//參數 IDS：存放某個選進的點集, 以便判斷是否爲 local independent dominating
set.
//參數 N：存放 r-Neighborhood 中的所有點.
//參數 nb_size：存放 r-Neighborhood 的 size.
//參數 size：被選進點集的 size.

int Neib( int node, int r, int adj[ ][NETWORK_SIZE+1], int N[ ], int wb[ ]);
//目的：找尋點 node 的 r-Neighborhood.
//需給傳入值的參數：node, r, adj, wb
//需做傳出值的參數：N
```

```
//傳回：node 的 r-Neighborhood 的 size.
//參數 node：由 node 這點出發，找尋這點的 r-Neighborhood.
//參數  r：同 check 函數.
//參數 adj：同 check 函數.
//參數 N：同 check 函數.
//參數 wb：當某點已經被其他點控制時，在找尋 neighborhood 的時候，就不需要
//     考慮該點, wb 就是用來記錄各個點是否已經被控制了.
//     wb 是 1 表示已經被控制, wb 是 0 表示尚未被控制.

void bruteforce(int node, int r, int adj[ ][NETWORK_SIZE+1], int IDS[ ],
                      int N[ ], int wb[ ], int &size);
//目的：以暴力法求出點 node 的 r-Neighborhood 的 minimum independent
//     dominating set (MIDS)以及此 MIDS 的 size.
//此 function 會呼叫 function Neib 及 function check
//需給傳入值的參數：node, r, adj, N, wb
//需做傳出值的參數：  IDS, size
//參數 node：同 Neib 函數.
//參數 r：同 check 函數.
//參數 adj：同 check 函數.
//此外 node 的 r-Neighborhood 的 MIDS 也用參數 IDS 傳出
//參數 IDS：若某個點集經由 function check 判斷出是 local independent
//     dominating set，則此點集將由 IDS 傳出.
//參數 N：同 check 函數.
//參數 wb：同 Neib 函數.
//參數 size：點 node 的 r-Neighborhood 的 MIDS 的 size，此值將傳出.

int GRAPH_num = 100; //設定跑 100 張圖.
int radius = 10;       //點跟點之間距離小於  radius，就會產生邊.
double epsi = 1.5;   //設定 PATS 中的誤差 epsi.
int threshold = 10;  //設定單次演算法執行時間假如超過 10 秒，即結束此回合.

int main(void)
{
     time_t t1, t2;
     time(&t1);       //將初始時間存進 t1 變數.
     int first, i, j, k, r, node, id, round;
     int wb[NETWORK_SIZE];
     int size, size2; //存放 local independent dominating set 的 size.
     FILE* fptr;
     char temp;
     char filename[7];
     char output[40];
     sprintf(output, "V_epsi=%.1f_radius=%d_size=%d_thresh=%d.txt",
                 epsi,radius,NETWORK_SIZE,threshold);
     int total_cardi = 0;  //最後 independent dominating set 的 size.

     //wb[NETWORK_SIZE]  紀錄各個點是否已經被控制了.

     ofstream outf;
     outf.open(output,ios::out);

     //以下步驟是執行 100 次不同圖的演算法
     for(round = 1; round <= GRAPH_num; round++)
     {
           int adj[NETWORK_SIZE][NETWORK_SIZE+1]; //記錄圖的連接矩陣.
           int N2[NETWORK_SIZE];                  //記錄點的(r+2)-Neighborhood.
           int N[NETWORK_SIZE];                   //紀錄點的  r-Neighborhood.
           int omega[NETWORK_SIZE];               //記錄我們求出的 independent
                                                  //dominating set.

           sprintf(filename, "%d.txt", round+ GRAPH_num);
           fptr = fopen( filename, "r");

           //以下步驟是以二維連接矩陣存取圖的結構.
           for ( i = 0; i < NETWORK_SIZE; i++ )
           {
                 wb[i] = 0;
                 for ( j = 0; j < NETWORK_SIZE+1; j++ )
                 {
                       temp = fgetc( fptr );
                       adj[i][j] = atoi( &temp );
                 }
           }
           fclose(fptr);
```

```
int index = 0;

//以下步驟是從 id 小的點去判斷該點是否已被控制.
for(i = 0; i < NETWORK_SIZE; i++)
{
    if (!wb[i])     //假如該點還沒被控制
    {
        int Nr_IDS[NETWORK_SIZE];
        int Nrtwo_IDS[NETWORK_SIZE];
        node = i;
        r = 0;

        //以下步驟為執行演算法內的 while 迴圈.
        do
        {
            bruteforce(node, r+2, adj, Nrtwo_IDS, N2, wb, size2);
            bruteforce(node, r, adj, Nr_IDS, N, wb, size);
            r = r+1;
        }
        while(size2 > (1+epsi)*size);

        if (!size2)
        {
            index = 0;
            break;
        }

        size2 = size2 + index;

        //以下步驟是將選定的點存進 omega 矩陣,
        //並將那些被 omega 矩陣控制的點標為 1.
        for(j = index; j < size2; j++)
        {
            omega[j] = Nrtwo_IDS[j-index];
            wb[omega[j]] = 1;
            for(k = 0; k < NETWORK_SIZE; k++)
```

```
                        if(adj[k][omega[j]])
                            wb[k] = 1;
                    }

                    index = size2;
                }
            }

            total_cardi = total_cardi + index;

        }

        (void) time(&t2);    //記錄演算法的終止時間.

        //以下爲 output 我們所求出的  the average cardinality of independent
        //dominating set 和整個程式的執行時間.
        outf<<(float)total_cardi/ GRAPH_num <<endl;
        outf<<(int)t2-t1<<"s"<<endl;
        outf.close( );

        system("pause");
        return 0;
}

int check(int adj[ ][NETWORK_SIZE+1], int IDS[ ], int N[ ], int nb_size, int s)
{
        int i, j, flag;

        //以下步驟是 check 選進的點集 IDS 是否 independent
        for(i = 0; i < s-1; i++)
            for(j = i+1; j < s; j++)
                if(adj[IDS [i]][ IDS [j]])
                    return 0;

        //以下步驟是 check 選進的點集 IDS 是否爲控制集
        for(i = 0; i < nb_size; i++)
        {
            flag = 0;

            for( j = 0; j < s; j++)
            {
                  if((IDS [j] == N[i]) || (adj[N[i]][ IDS [j]] == 1))
                  {
                        flag = 1;
                        break;
                  }
            }

            if (!flag)
                  return 0;
      }

      return 1;
}

int Neib(int node, int r, int adj[ ][NETWORK_SIZE+1], int N[ ], int wb[ ])
{
        int id = 1;
        int current = 1;
        int i,j,k,temp;
        int node_wb[NETWORK_SIZE];
        N[0] = node;

        //以下步驟是以 node_wb 矩陣存取該點是否被選過，避免重複選到
        for(i = 0; i < NETWORK_SIZE; i++)
              node_wb[i] = 0;

        node_wb[node] = 1;

        //以下步驟求出 1-Neighborhood
        for(i = 0; i < NETWORK_SIZE; i++)
        {
              if (((adj[N[0]][i]) && (!node_wb[i])) && (!wb[i]))
              {
                N[id] = i;
                node_wb[i] = 1;
                id++;
            }
        }

        //以下步驟求出 r-Neighborhood
        for(k = 1; k < r; k++)
        {
            temp = id;

            for(j = current; j < temp ; j++)
            {
                for(i = 0; i < NETWORK_SIZE; i++)
                {
                    if (((adj[N[j]][i]) && (!node_wb[i])) && (!wb[i]))
                    {
                        N[id] = i;
                        node_wb[i] = 1;
                        id++;
                    }
                }
            }

            current = temp;
        }

        return id;
}

void bruteforce(int node, int r, int adj[ ][NETWORK_SIZE+1],
                int IDS[ ], int N[ ], int wb[ ], int &size)
{
    int first, i, s, nb_size;
    time_t t3, t4;
    (void) time(&t3);    //記錄此暴力法的初始時間.
    int b[NETWORK_SIZE];
```

```
nb_size = Neib(node,r,adj,N,wb);

if (nb_size == 1)
{
        IDS[0] = N[0];
        size = 1;
        return;
}

//以下步驟以暴力法求出最佳解.
for(s = 1; s < nb_size; s++)
{
        (void) time(&t4);    //記錄此時時間.

        //以下步驟是判斷執行時間是否超過 10 秒,假如超過就回傳 0.
        if ((int)t4-t3 > threshold)
        {
                size = 0;
                return ;
        }

        int flag = 0;

        for( i = 0; i < s; i++)
        {
                b[i] = i;
                IDS[i] = N[i];
        }

        if (check(adj, IDS, N, nb_size, s))
        {
                size = s;
                return;
        }

        while(!flag)
        {
            flag = 1;

            for( i = s-1; i >= 0; i--)
            {
                if (b[i] != (nb_size – s + i))
                {
                    first = i;
                    flag = 0;
                    break;
                }
            }

            if (!flag)
            {
                b[first] = b[first]+1;
                IDS [first] = N[b[first]];

                for( i = first+1; i < s; i++)
                {
                    b[i] = b[i-1]+1;
                    IDS [i] = N[b[i]];
                }

                if (check(adj, IDS, N, nb_size, s))
                {
                    size = s;
                    return;
                }
            }
        }
    }
}
```