Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Local construction of dominating set

DIPLOMOVÁ PRÁCA

Bc. Jozef Spišiak

BRATISLAVA 2011

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Local construction of dominating set

DIPLOMOVÁ PRÁCA

Evidenčné číslo: 401db6ff-2c1f-4910-85be-d6720dc08906
Študijný program: Informatika
Študijný odbor: 9.2.1. Informatika
Školiace pracovisko: Oddelenie Informatiky, MÚ, Slovenská akadémia vied
Školiteľ: RNDr. Stefan Dobrev, PhD.

Bc. Jozef Spišiak
Bratislava 2011

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Jozef Spišiak
Študijný program:	informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	9.2.1. informatika
Typ záverečnej práce:	diplomová
Jazyk záverečnej práce:	anglický

Názov: Local construction of dominating set

Cieľ: Analysis of local algorithm and its dependency from input parameters

Vedúci: RNDr. Štefan Dobrev, PhD.

Dátum zadania: 20.11.2009

Dátum schválenia: 18.02.2011

[illegible]
prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

[illegible]
....
študent

[illegible]
....
vedúci

Čestné prehlásenie

Vyhlasujem, že som diplomovú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

Bratislava 6. 5. 2011 ....

Vlastnoručný podpis

Poďakovanie

Ďakujem vedúcemu diplomovej práce RNDr. Stefanovi Dobrevovi, PhD. za odporučenie tejto zaujímavej témy, za odbornú pomoc a pripomienky pri jej vypracovávaní.

Abstrakt

SPIŠIAK, Jozef: Local construction of dominating set [Diplomová práca]
Univerzita Komenského v Bratislave.
Fakulta matematiky, fyziky a informatiky.
Vedúci: RNDr. Stefan Dobrev, PhD. Bratislava : UK, 2011, 48 s.

Táto práca sa venuje problematike lokálnej konštrukcie množiny nazývanej dominantná. Jej jadro tvorí lokálny algoritmus na nájdenie tejto množiny na grafe (sieti). Práca analyzuje tento algoritmus vzhľadom na jeho vstupné parametre. Štatisticky vyhodnocuje efektivitu algoritmu a porovnáva jeho jednotlivé varianty (modifikácie). V siedmej kapitole ukazuje optimalizáciu úpravou algoritmu generujúcu ešte menšiu dominantnú množinu. Vizualizuje daný algoritmus a poskytuje čitateľovi v rámci príloh zdrojové kódy programov, vytvorených pre účely tejto práce. Taktiež obsahuje ukážky v programovacom jazyku C++. Porovnáva použitie algoritmov pre rôzne hustoty siete.

Kľúčové slová: dominantná množina, bezdrôtové siete, grafy, NP-úplný problém, lokálny algoritmus

Abstract

SPIŠIAK, Jozef: Local construction of dominating set [Master thesis]
Komenius university in Bratislava.
Faculty of mathematics, physics and informatics.
Supervisor: RNDr. Stefan Dobrev, PhD. Bratislava : UK, 2011, 48 p.

This thesis handles the local construction of dominating set problem. It's core is being constructed on local algorithm for finding this set on graph (network), analyzing the algorithm input parameters. Statistical research determines effectiveness of algorithm and its variants (modifications). Chapter 7 shows the improvement for this algorithm generating smaller dominating set. It visualizes this algorithm and also provides reader with helpful source codes made for this work. It also contains examples written in programming language C++. It shows which algorithms are best for using on networks, depending on the density of network.

Keywords: dominating set, graph theory, NP-complete problem, local algorithm, wireless networks

Contents

Introduction

This diploma thesis handles the problem known as Local construction of dominating set. It is recommended to know basics of graph theory. The first chapter formulates basic definitions and defines the working area around the problem. It also discusses my motivation in this topic. The second chapter is mainly about the topic of computational complexity of known solutions. Starting with chapter, the thesis describes my work in this field. It begins with finding the optimal parameters for presented algorithm. First parameter is class permutation. The next chapter presents graphical program for visualizing. The 6th chapter is about choosing the right dominator inside small area (cluster). Various methods are being shown, but in the following chapter, a new improvement for this algorithm is given. Chapter 8 handles documentation for included DVD and code snippets. Chapter 9 summarizes results for dominating on graphs the last chapter is global summary.

Chapter 1

Introduction to topic

First of all, what is dominating set? “Consider a graph G(V,E) with vertex set V and edge set E. A subset S of V is called dominating set if every vertex of G is either in S or adjacent to a vertex in S.” [CDF^{+}08] The domination number γ(G) is the number of vertices in a smallest dominating set for G. On every set can be found at least one dominating set, because the set of all vertices is dominating according the definition. The problem is to find the smallest dominating set, or in other words, for given K and graph G answer, whether the inequality $\gamma(G) <= K$ is true or not. Some examples of minimal dominating sets (the blue vertices are members of dominating set):

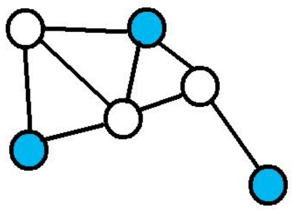


Figure 1.1: Dominating set example

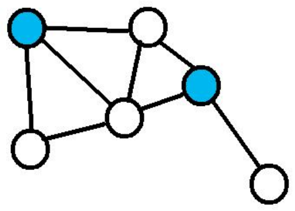


Figure 1.2: Dominating set example (minimal)

1.1 Formulation of problem

Because the problem of minimal domination of set is NP-hard to solve (will be discussed in chapter /refchap:complexity), this thesis will be analyzing only a sub problem, oriented on special class of graphs, called unit disk graphs (UDG). A UDG in graph theory is defined by vertices V and radius r. Edge is defined as intersection of two circles with radius r between vertices in the middle of these circles containing these vertices. On these graphs, the problem is easier to solve, because we know the minimal and maximal distances between adjacent vertices depending on shortest way. However still "The MDS problem is NP-hard, even on unit disk graphs when a geometric representation is given." [NH04] //

Therefore formulation of problem is: For given graph G $\in$ UDG with location aware vertices, find the best approximation of minimal dominating set using local algorithm.

1.2 Known facts

Global approach to this problem brings following boundaries. Approximable within $1 + \log |V|$ since the problem is a special instance of Minimum set cover. [Joh73]. Not approximable within $c \log |V|$, for some c > 0. [RS97]

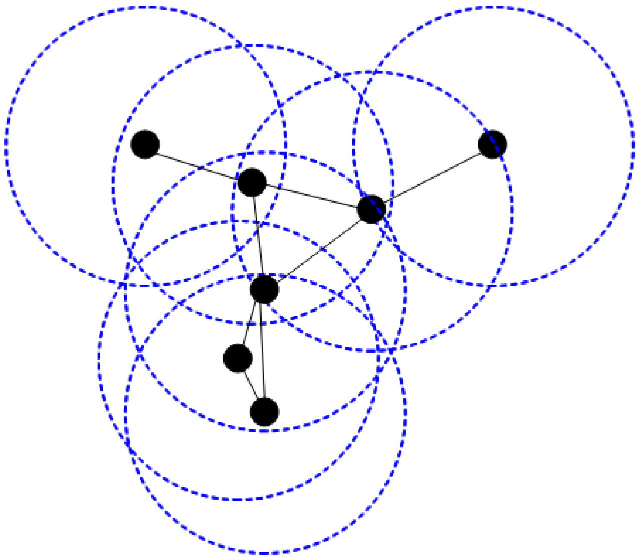


Figure 1.3: Unit disc graph example (transmission ranges shown as blue circles)

1.3 Motivation

The domination problem is known from the 1950s onward, but the research on it really begun in mid-1970s. UDG graph models ad-hoc wireless communication networks (if you have same ranged access points (AP) or take R = minimal range on access point/2). “Wireless ad hoc networks can be flexibly and quickly deployed for many applications such as automated battlefield operations, search and rescue, and disaster relief. Unlike wired networks or cellular networks, no physical backbone infrastructure is installed in wireless ad hoc networks. A communication session is achieved either through a single-hop radio transmission if the communication parties are close enough, or through relaying by intermediate nodes otherwise.” [AjWF02] In these networks one of efficient ways of broadcasting and retrieving information is finding the minimal dominating set and using it as a backbone for communication. It is especially useful in networks with high density. Those can be found in high urbanized areas or special industries. For example in army where intelligent weapons and robots need to communicate with each other. Another example can be found in nanotechnologies, where you need to gather pieces of information from multiple robots on small area. Therefore it is useful to push forward and gain new possibilities in this area.

Chapter 2

Computational complexity of known solutions

In this chapter will be presented the computational complexity of given problem and known solutions.

2.1 Finding minimal dominating set in graph

This problem belongs to class of NP-complete problems. It can be shown, that it can be transformed in minimum set cover problem. From dominating set problem to set covering problem: For graph G=(V,E) with |V| = n, we need to construct set cover instance (S,U). Let a universe U = V and family of subsets S = $\{S_1, S_2, ..., S_n\}$ be made, where S_v consists of vertex v and all vertices adjacent to v in G. Now for given solution of minimal set covering problem S, all the vertices will also be the members of minimal dominating set in graph G, because of construction of set S. In set S are all vertices and each member of set S has at least one vertex in dominating set.

From set covering problem to dominating set problem: Beginning with covering problem instance (S,U), where U is universe and S = $\{S_i, i \in I\}$, there will be constructed graph G=(V,E), where V = $U \cup I$ and edges are between each pair of members I and {i,u} where $i \in I$ and $u \in S_i$. The graph G becomes a split graph: I is clique and U is independent set. Now for given minimal cover set C, the dominating set is represented on I vertices of this graph, with indexes of our cover set solution C.

2.2 Finding minimal dominating set in UDG

In UDG class the problem is still NP-hard. But in this case, there is a local approximation algorithm (polynomial time approximation scheme), which outputs in polynomial time dominating set, that has at most k times more vertexes as minimal. The word local means that each vertex has one instance of this algorithm running and does not know the whole network topology. It only operates within its small area. “Therefore, the local approach is the most desirable to support scalable design through localized maintenance in a dynamic environment (also called locality).” [WDY08] This algorithm then gives information, whether the vertex is part of dominating set or is not. One of these algorithms is described below.
The idea behind this algorithm is that we can divide the whole plane into small areas (clusters) and compute dominating sets on them without losing too much efficiency, when we join them into one dominating set. “By clustering the entire network, one can decrease the size of the problem into small sized clusters.” [CED06] For this, the vertexes need to know their coordinates on plane, so they can decide to which area they belong to. Here is also used the fact, that in UDG are edges have maximum length 2R, therefore the minimal dominating set in sub graph is relatively good approximation for part of global minimal dominating set.
The tiling of plane should be regular and should divide it into right sized areas. Therefore this algorithm is using 12-hexagon tiling of plane with the one hexagon side of length R, as shown on figure 2.1.
The numbers in the hexagons represents classes of hexagon. They are important in next lemma.

Lemma 2.2.1 *In the tilling of the plane given before any two vertexes on the plane, that are of the same class, but belong to two different hexagons, are at Euclidean distance greater than 2. Moreover, given the coordinates of a vertex P in the plane, one can determine the class number of P using a calculation of constant cost.*

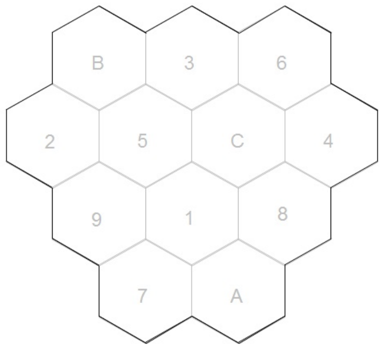


Figure 2.1: one piece of 12-hexagon plane tiling

Proof Take a look at these two figures above (2.2 & 2.3). Without loss of generality pick hexagons with classes 1. The shortest distance between two of these hexagons goes through two sides and one diagonal of the hexagon. That makes according to figure 3 distance of 4R, which is in speaking of maximal edge distance 2R from definition, Euclidean distance 2. Due to the symmetry of the tiling used, the distance requirements just proved for class 1 hexagons apply to any other class number. Given the coordinates to the vertex, the class number can be calculated without any interaction with other vertexes, therefore consider this calculation as constant costing. ∎

Remark If all vertexes are aware of tiling being used, each of them can calculate its own class number that it belongs to. In following, this information is used as given to the vertexes.

In each hexagon, all vertexes can be dominated by one vertex (they are all in range), so the local algorithm needs to find it. The algorithm makes following steps:

1. Find all neighbours and determine their coordinates and classes.

2. If you have class 1, the nearest vertex to the center of your hexagon belongs to dominating set. Continue to step 5.

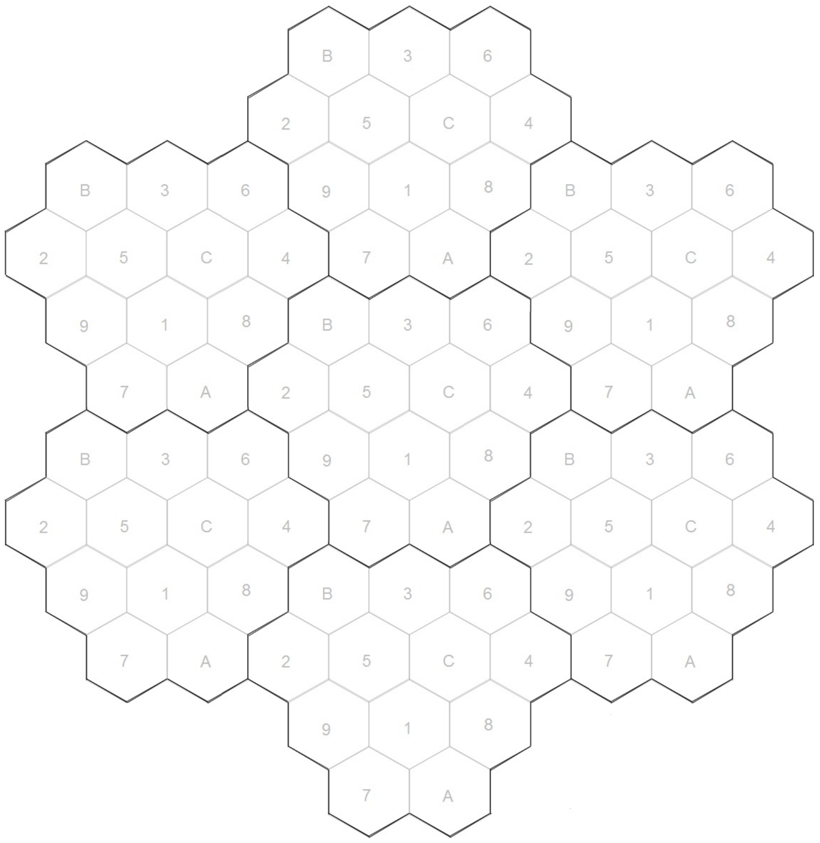


Figure 2.2: Tiling shown on more pieces

3. If there is a vertex in your hexagon that has not neighbours with lower classes, then the nearest to the center of these vertices is designed dominator. Continue to step 5.

4. If you have a lower class neighbours, wait until you get messages from them about finishing choosing dominator and find if you are already dominated. Get this information from all vertices in your hexagon. The nearest vertex to the center from those, who are not dominated, is designed dominator, if exists.

5. Inform all your neighbours about dominator in your hexagon.

6. Finished.

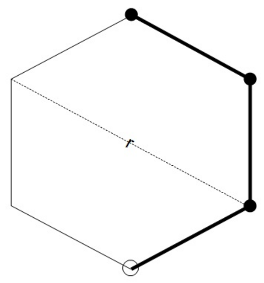


Figure 2.3: One hexagon

Next two lemmas are discussing domination and independence of output from this algorithm.

Lemma 2.2.2 *Each vertex of G after running algorithm dominated or dominator. Therefore set of dominators is dominating set.*

Proof Let X not be a member of D. If it has class 1, then it was dominated in step 2 of algorithm, because distance to the nearest vertex to center is less than r. Also if it has no neighbours with lower classes, it has been dominated in step 3. Since the hexagon has diameter r, it must have been dominated in step 4, if it had neighbours, that did not dominate him. Therefore this proof is complete. ▮

Lemma 2.2.3 *D is independent set. In other words, the Euclidean distance between two members of D is greater than 1.*

Proof To have Euclidean distance 1 means they are adjancent. In step 2, there are not any 2 adjacent vertices designed as dominators. Also in step 3, there cannot be any added, because these vertices are dominated only by the one closest to the center. In step 4, if the vertex is dominated, it is not added as designed dominator and if it is not, it becomes dominator or dominated by the dominator closest to middle. ▮

Theorem 2.2.4 *Let G be unit disk graph and let D be the set of dominators, computed by algorithm. D is independent and dominating set of graph G. Let*

D^* be a minimum dominating set. Competitive ratio of this algorithm is 5 or in other words $|D|/|D^*| \leq 5$.

Proof According to lemmas 2.2.2 and 2.2.3, D is both, dominating and independent. Now let's take vertex X as a vertex of D*, that does not belong to D. If does not exist, than D = D*, because D* is minimal. This vertex X must be covered by vertex from D, because D is dominating. Take two vertices X_1 and X_2, that are dominating X. The angle $\angle X_1 X X_2$ must be greater than $\pi/2$, otherwise it is in contradiction with lemma 2.2.3, because Euclidean distance between X_1 and X_2 would be 1. Therefore X can be dominated by maximum 5 vertices of set D. This implies that |D|/|D*| ≤ 5. ∎

"If the vertices of degree 5 are placed in hexagons of high class number, the dominating set D computed by the algorithm includes only the vertices of degree 1 and 2 while the optimum one consists of the vertices of degree 5." [**?**]

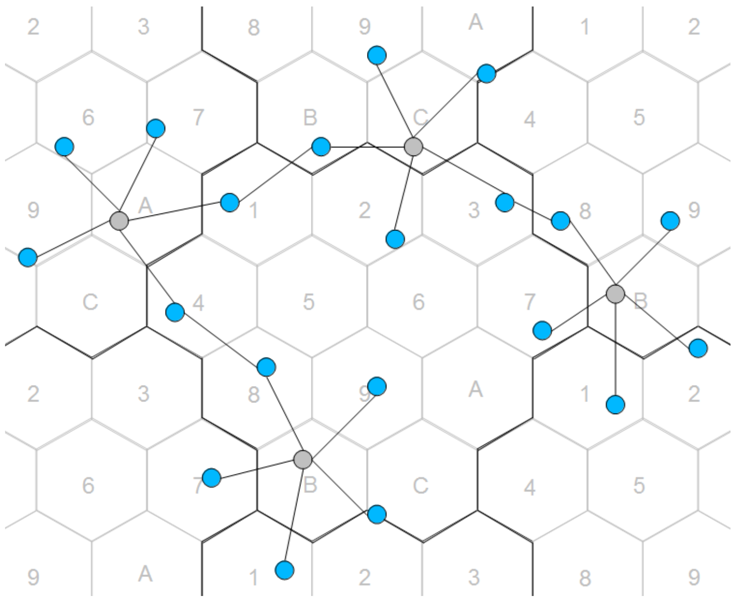


Figure 2.4: example of a set of vertexes, where the ratio between the minimal D* and D is greater than 4

2.3 Questions discussed

The main question is how effective is local algorithm presented in next chapter. The worst case shown is showing the dominating set can be 5 times larger than minimal, but what about average case? Does it depend on class permutation or density of network? How to choose dominator inside the class?

Chapter 3

What to measure? How to make a good statistics

Following work is computer-based statistical data analysis. Therefore it is important to know which the key factors for measurement of algorithm efficiency. The main factor is naturally size of dominating set, but we can also include others.

3.1 Size of dominating set

First of all measure the size of dominating set. For statistical purposes is this work using mean from more measurements, to assure better accuracy. Mean is defined as sum of values divided by count of values: $\frac{\sum_{i=0}^{n} D_i}{n}$. Standard deviation defined as dispersion from mean is also measured. It shows how much are results different from mean. Formal definition: $\sqrt{\frac{\sum_{i=0}^{n}(D_i - Mean)}{n}}$

3.2 Area of coverage

Another factor of efficiency is area of coverage. It is being calculated from dominating set nodes by dividing sum of area the nodes have covered by area (M) on which the dominating set is being computed. The final number is being shown as percentage of the whole area. Following code executes this functionality. The idea behind these lines of code is: The area covered by one dominator is PI * transmission range squared. Therefore it uses cardinality of dominating set and multiplies it by area covered by one dominator. However, it must subtract the area, that is being covered outside of M and area, which is being covered by more than one dominator. This is represented by segments, that are computed in for cycle. Approximation is used, where the computation of overlapping 3 and more coverage circles is being ignored, as the set is being built independent. For more documentation to the variables and constants, see documentation section. **??**

```
segments = 0;
for (i=0; i<N; i++)
{
  if (sur[i][3] == STATE_LEADER)
  {
  for (j = i+1; j<N; j++)
  {
    distanc = sqrt(
                pow(sur[i][0]-sur[j][0],2)+
                pow(sur[i][1]-sur[j][1],2)
                );
    if (sur[j][3] == STATE_LEADER && distanc<4*A)
    {
      segments += 8*A*A*acos(distanc/(4*A))
        -
          distanc
          *
          sqrt(
          4*A*(-distanc/2 + 2*A)-
            pow((-distanc/2 + 2*A),2)
          );
    }
    }
    distanc = 0;
    if (sur[i][0] < 2*A) distanc = sur[i][0] * 2;
    if (sur[i][0] > MAX-2*A) distanc = (MAX-sur[i][0])*2;
    if (sur[i][1] < 2*A) distanc = sur[i][1] * 2;
    if (sur[i][1] > MAX-2*A) distanc = (MAX-sur[i][1])*2;
    if (distanc > 0)
      segments += 2*A*A*acos(distanc/(4*A))
        -
          distanc/2
          *
          sqrt(
          4*A*(-distanc/2 + 2*A)-
            pow((-distanc/2 + 2*A),2)
          );
    }
}
area += (l*3.14159265358979323846264338327950*
4*pow(A,2) - segments)/pow(MAX,2);
```

Chapter 4

Permutations of classes

Algorithm presented in chapter 2 uses only one permutation of classes within 12 hexagon area and does not answer the question whether this permutation is the best. This chapter resolves question whether the size of dominating set is different when using other permutations of classes used in algorithm or not. If there are differences, then the natural question is: Which permutation gives us in most cases smallest dominating set?

The most cases is used because of following lemma:

Lemma 4.0.1 *For every two permutations exists a graph G, which cardinality of its dominating set using first permutation is lower than cardinality of its dominating set created using second permutation.*

Proof For construction of this graph only 3 nodes are needed. Find a lowest (by number of first) cluster difference between permutations. Place into lowest cluster one node, other two place inside higher classes in first permutation, so there the two placed nodes would have no edge between selves, but still will be connected to first node. Dominating graph given by algorithm has in fist case cardinality 1 and in second 2, therefore we constructed this graph G.&

4.1 Differences between permutations

First of all, the algorithm phase1 described in chapter 8 was written, to generate random input sets. Using this program, the input set of input groups was generated. These were on square area with side 10000, using transmission range 1000 and number of nodes ascending from 300 to 2900 by 100. Each group had 2000 random generated files for statistical purposes. Using algorithm phase2b, the local construction of dominating set is being simulated and mean is being calculated for groups. Following graph shows results of this experiment. The values are clearly smaller for some permuta-

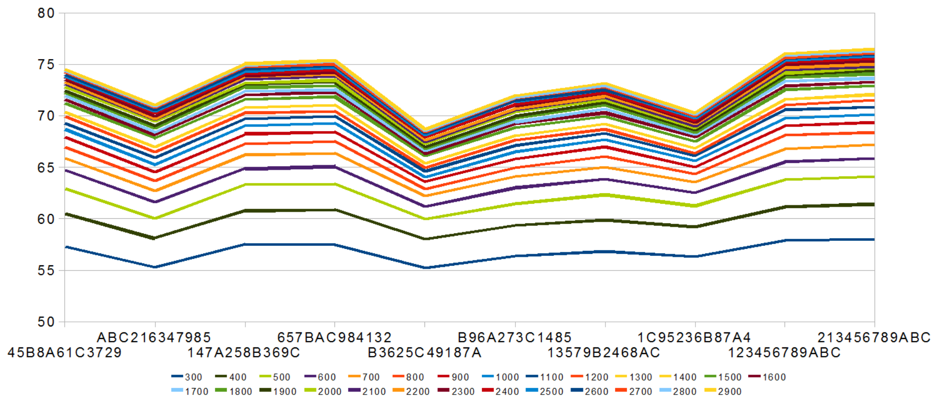


Figure 4.1: Dependency of average size of dominating set on used permutation in algorithm

tions than for the others, therefore the hypothesis is:

Hypothesis 4.1.1 *For uniform random distributed input some permutations of classes are better than others.*

To support this hypothesis, following graph is showing standard deviation. The difference in mean size of dominating set between class permutations is bigger than deviation in some cases: e.g. 1234566789ABC and B3625C49187A.

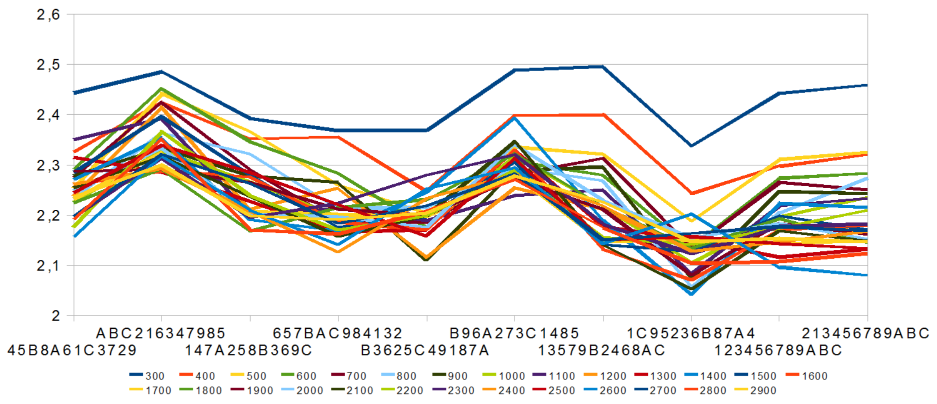


Figure 4.2: Dependency of standard deviation size of dominating set on used permutation in algorithm

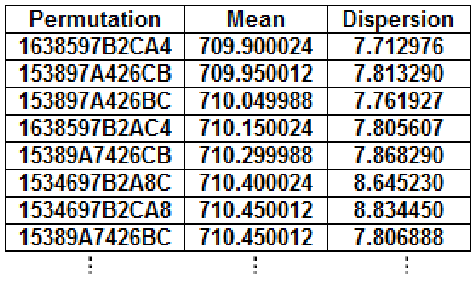
4.2 Best permutation

To prove the hypothesis the best permutation should be found, in meaning of lowest mean of dominating set.

Definition Transposition is a permutation which exchanges two classes and keeps all others fixed.

Definition Two permutations of classes are isomorphic, if an infinite area divided into clusters with classes using first is equal to area divided into clusters with classes of second permutation.

Therefore to support the search table showing in figure 4.3 was created with algorithm Generate1. This algorithm generates all possible permutations using classic backtrack algorithm, for searching all the options that are not isomorphic. To do so, it allows only transpositions of adjacent clusters, excluding first cluster with class 1. Finally it writes out permutations ascending, ignoring lower permutations than last outputted. The output around 16000 permutations is a good set of permutations to test, what are the best permutations. Using algorithm phase2be the results were found on group of inputs. The following figure is only a part of original table, sorted ascending by mean. The top 8 permutations have the same position of clus-



Permutation	Mean	Dispersion
1638597B2CA4	709.900024	7.712976
153897A426CB	709.950012	7.813290
153897A426BC	710.049988	7.761927
1638597B2AC4	710.150024	7.805607
15389A7426CB	710.299988	7.868290
1534697B2A8C	710.400024	8.645230
1534697B2CA8	710.450012	8.834450
15389A7426BC	710.450012	7.806888
⋮	⋮	⋮

Figure 4.3: Top part of table with permutations sorted ascending by mean

ters with classes 1, 2 and 3. Other top permutations have only minimal variation. Next picture is visualizing this result. The set of best permutations is isomorphic with permutation $1X_1 3X_2 X_3 X_4 X_5 X_6 2X_7 X_8 X_9$, where $X_1 \neq X_2 \neq X_3 \neq X_4 \neq X_5 \neq X_6 \neq X_7 \neq X_8 \neq X_9 \in \{$4,5,6,7,8,9,A,B,C$\}$.

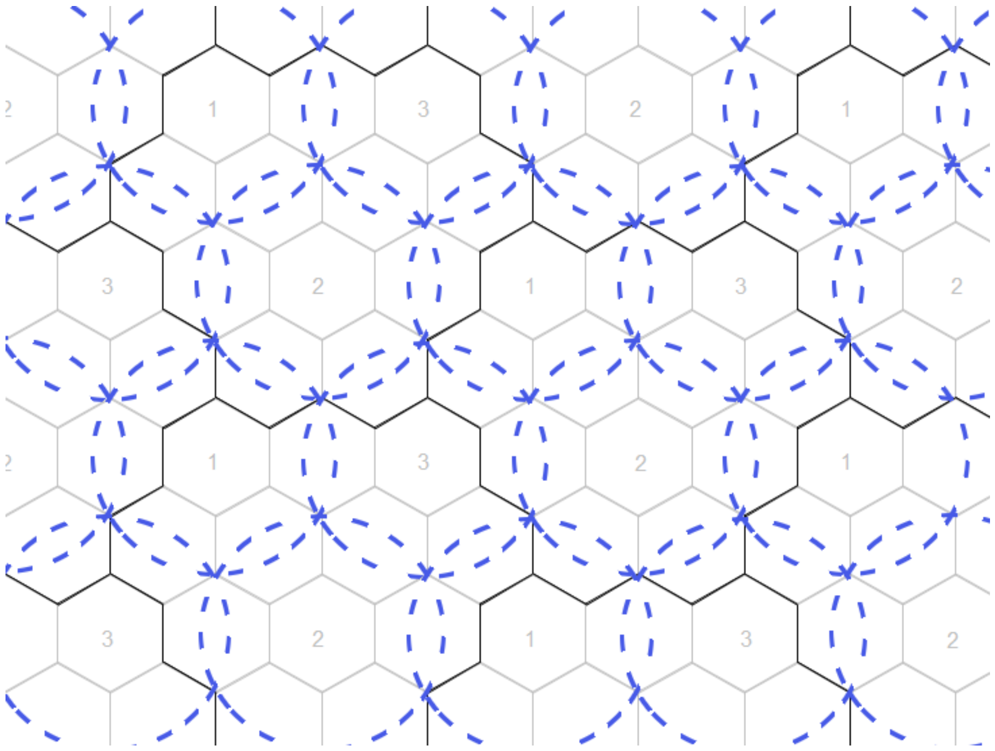


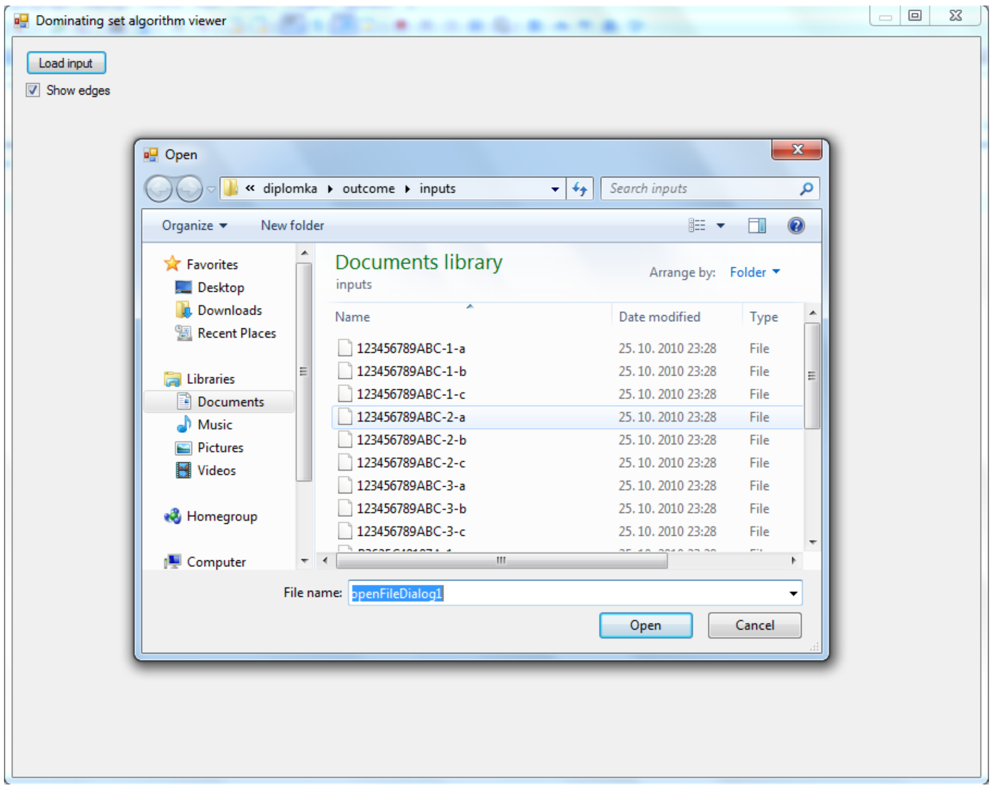
Figure 4.4: Clusters with classes 1, 2 and 3. Those have best permutations in common. Blue dashed circles are show transmission range from middle of cluster.

Chapter 5

Algorithm visualization

To help understand, how does the algorithm works, an application for visualization of this algorithm has been made in C#. When user launches Viewer, he can choose to load input or hide edges. Hiding edges is available also when input is loaded. Algorithms without batch processing to generate input for single set visualization (documented in chapter 8). Window can be resized to any size, the canvas for drawing is resized in next redraw.

After the input file is loaded, the visualization begins. Red nodes are dominators and gray are being dominated. Actual step in domination is being shown with full circles. Animation can be paused with clicking anywhere on visualization canvas. Visualization for one class is slowed down to dominating one node per second, to ensure enough time for user to track changes. In real application, the nodes in one class are being dominated in random order in random times.
In the end all nodes are dominated or dominators. It is possible to search for non-optimal sections of dominating set and improvements in algorithm.



Name	Date modified	Type
123456789ABC-1-a	25. 10. 2010 23:28	File
123456789ABC-1-b	25. 10. 2010 23:28	File
123456789ABC-1-c	25. 10. 2010 23:28	File
123456789ABC-2-a	25. 10. 2010 23:28	File
123456789ABC-2-b	25. 10. 2010 23:28	File
123456789ABC-2-c	25. 10. 2010 23:28	File
123456789ABC-3-a	25. 10. 2010 23:28	File
123456789ABC-3-b	25. 10. 2010 23:28	File
123456789ABC-3-c	25. 10. 2010 23:28	File

File name: openFileDialog1

Figure 5.1: Loading input for Viewer

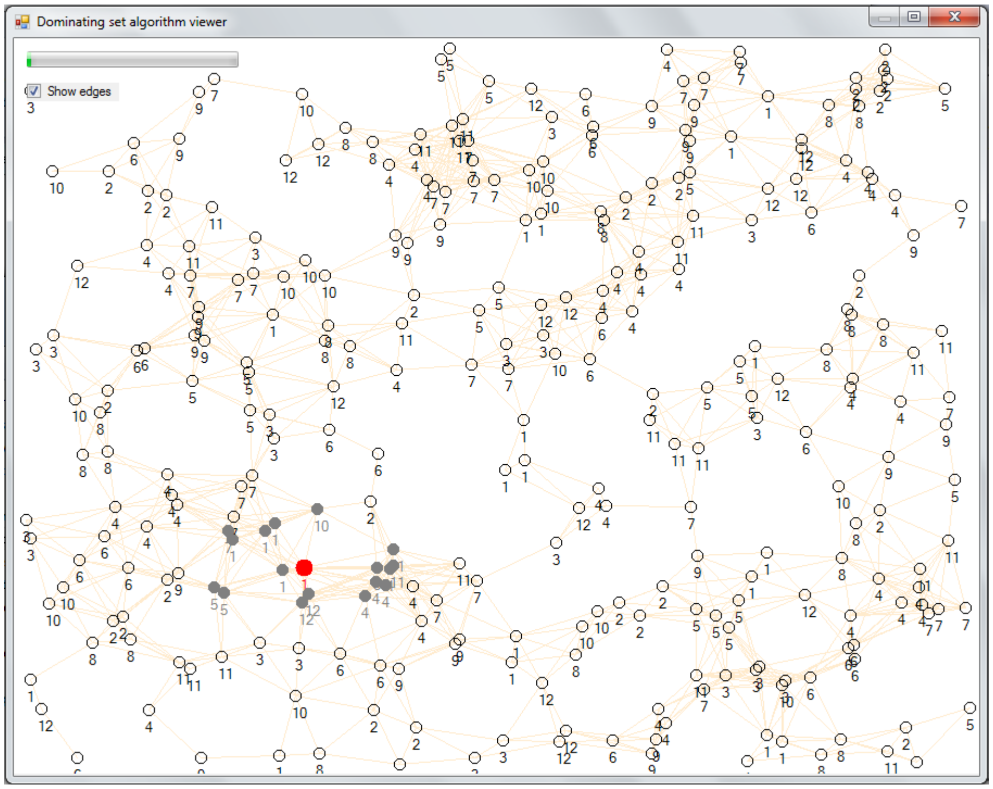


Figure 5.2: Begin visualization with Viewer

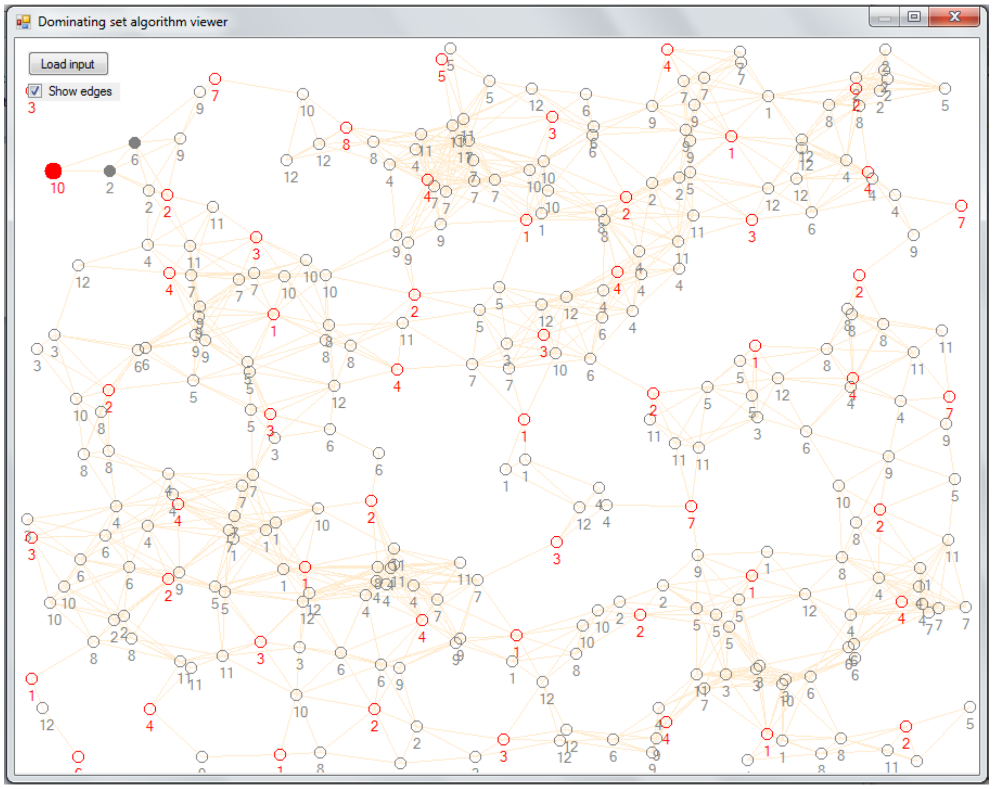


Figure 5.3: Visualization ended

Chapter 6

Choosing the dominator

Choosing the dominator within cluster with class is also an important factor on size of dominating set. There are several ways of doing this. First and fastest option is choosing it at random.

6.1 Random dominator

This can be implemented easily. Every unassigned node (is not dominator or being dominated by one), when searching for dominator in its class, generates random number and broadcasts it to its neighbours sharing class with it. The node with the highest number is the dominator and broadcasts this information. Simulating this behavior on area 20 000 x 20 000 containing 500-4500 nodes with transmission range 500, following graph showing mean and standard deviation of 2000 input files for each density was made.

6.2 Most neighbours

This option considers number of potential dominators. Each unassigned node broadcasts its neighbours count to nodes sharing its class number. The node with highest count is being set as dominator. This way it dominates most edges, therefore gaining potentially smaller dominating set. Simulating on the same input files as previous case following graph was created.

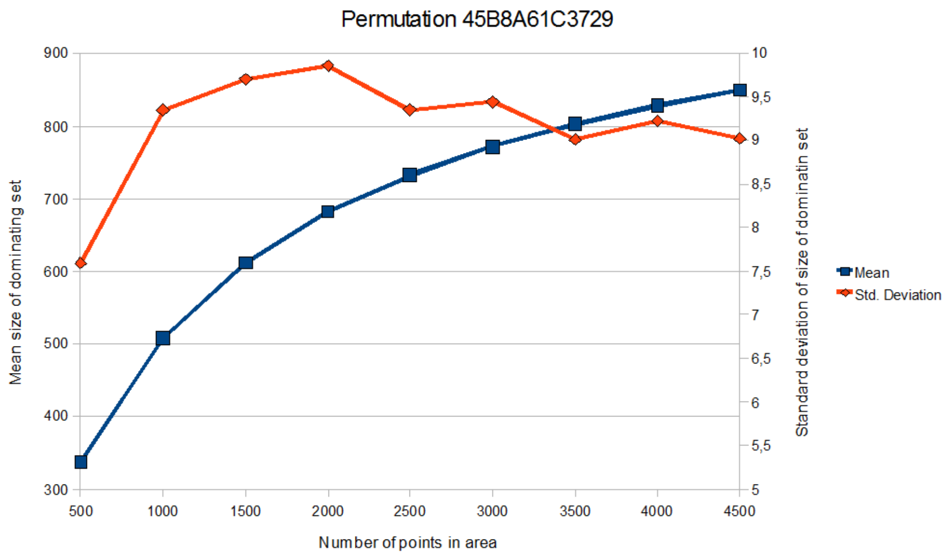


Figure 6.1: Graph for random dominator within class

6.3 Most unassigned neighbours

Previous option has one bad case, when there was big density within range of some node, the algorithm set as dominators nodes near each other. To avoid this, only unassigned neighbours count. Each node asks its neighbour, if it is being dominated or not (it cannot be dominator, because then the node asking is being dominated). Then it counts unassigned neighbours and broadcasts the number to see if it is being dominator of this cluster.

6.4 Closest to center

When modelling on high density networks, the best permutation covers with dominators from first 3 classes the whole area, if they are in the middle of cluster. Thus graph with mean and standard deviation was made on higher density set of nodes from 5000 to 40000. Also one of the best permutations was used. In next chapter 7 it will be compared with others.

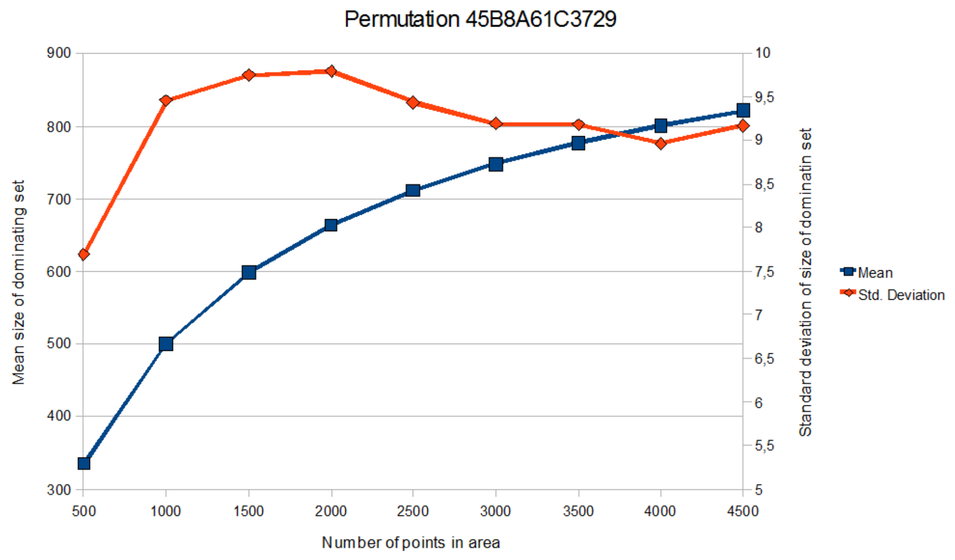


Figure 6.2: Graph for most neighbours dominator

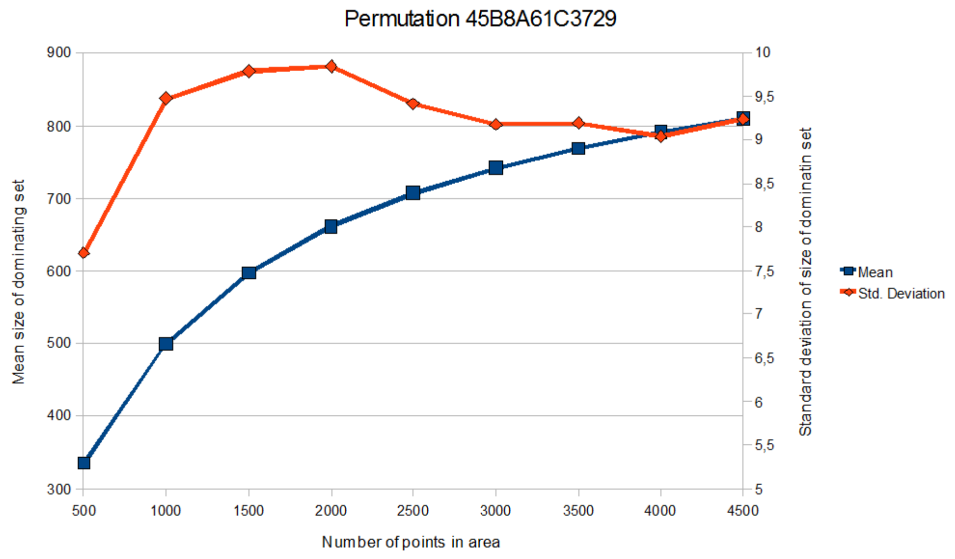


Figure 6.3: Graph for most non-assigned neighbours dominator

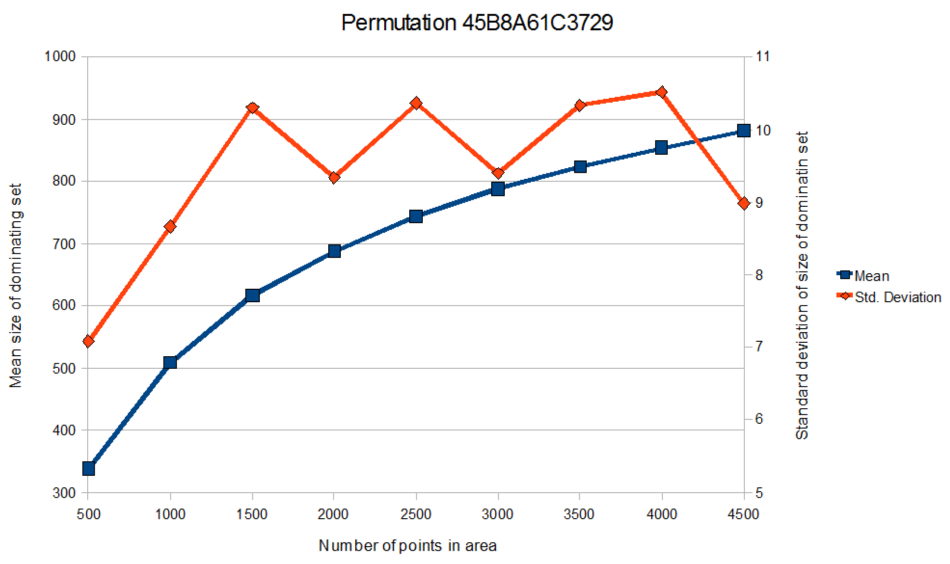


Figure 6.4: Graph for centered in cluster dominator

Chapter 7

Improving the algorithm

Algorithm has worst case presented in chapter 2 where the 5 time larger than minimal dominating set is being computed. This is direct result of dividing area into hexagons with diameter of transmission range. It is class permutation independent. Therefore the new approach must be involved, to improve this worst case.

7.1 Looking behind the borderline

When selecting the dominator inside cluster with class, the surrounding clusters can be searched for dominating the non-assigned nodes inside this class. This eliminates the worst case completely, giving minimal dominating set for this solution. However, to be able to select dominator parallel across all clusters with the same class, the number of classes must be raised to 19. Otherwise there could be 2 dominators selected in one step, which could see each other inside its transmission range. That would result to algorithm, which is not deterministic and would not likely produce best solutions. Based on the knowledge of constructing the best permutation for 12 classes, the best permutation for 19 classes is being constructed below.

Following graph shows values in statistical testing of this new implementation.

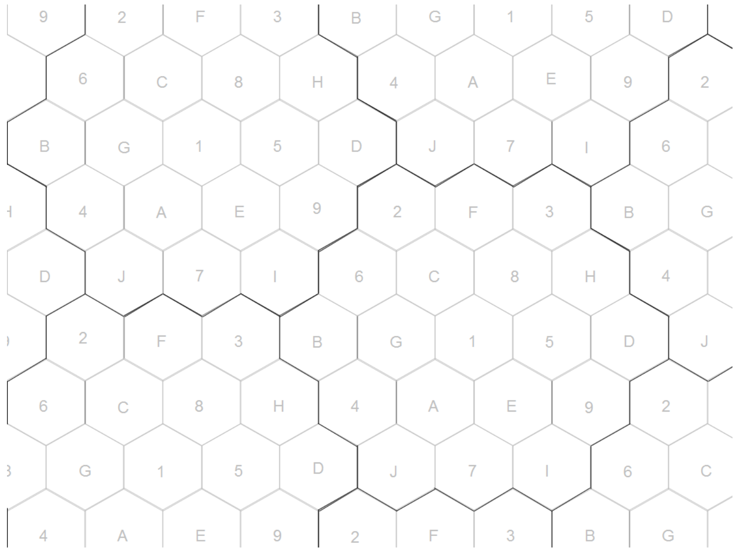


Figure 7.1: Figure representing dividing area into clustered numbered with 19 classes - best permutation shown

It seems to be pretty good and also all values are better than previous variations. However it can be also improved as shown in the next section.

7.2 Improvement 2

The previous algorithm had worst case presented on figures 7.3 and 7.4. The minimal dominating set is smaller in this case 1 node smaller as shown, therefore the algorithm could generate dominating set 4/3 greater. This could be improved by looking for dominator only from one node. First unassigned node from cluster is being selected (e.g. lowest ID) and it is looking for best dominator. Therefore, there is a chance for finding more optimal dominating set. As a matter of fact, the statistical results prove it to be better.

However the worst case, the improved modifications are solving is available only in low density networks, therefore there will be comparison in Results section, comparing these variations of algorithm.
Other improvements were tried out, but the resulting dominating sets were

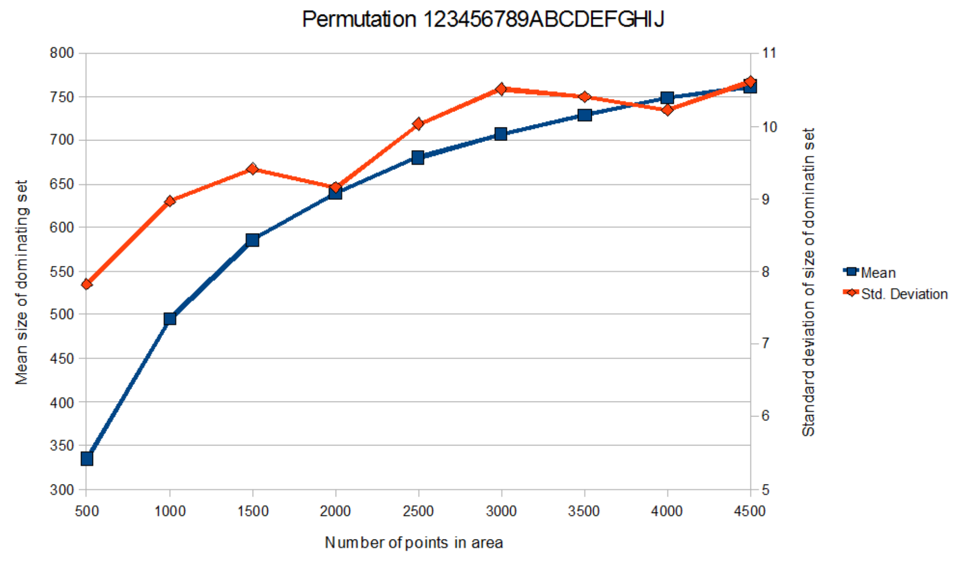


Figure 7.2: Graph for Improvement 1

much larger in comparison to these.

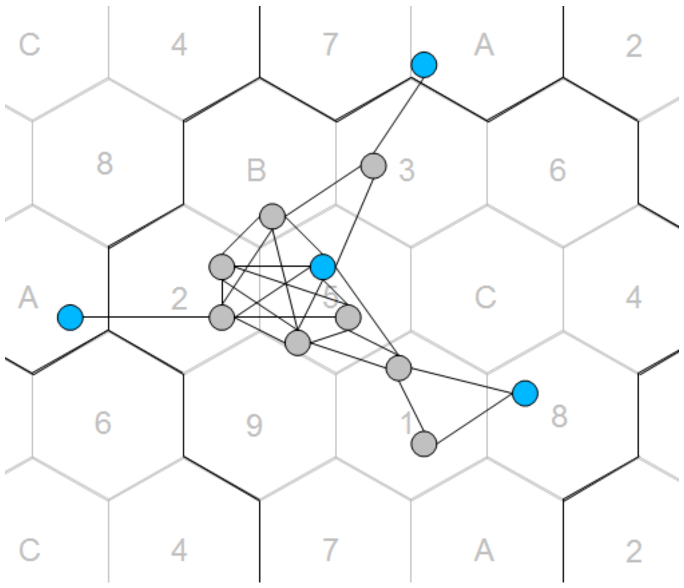


Figure 7.3: Dominating with Improvement 1

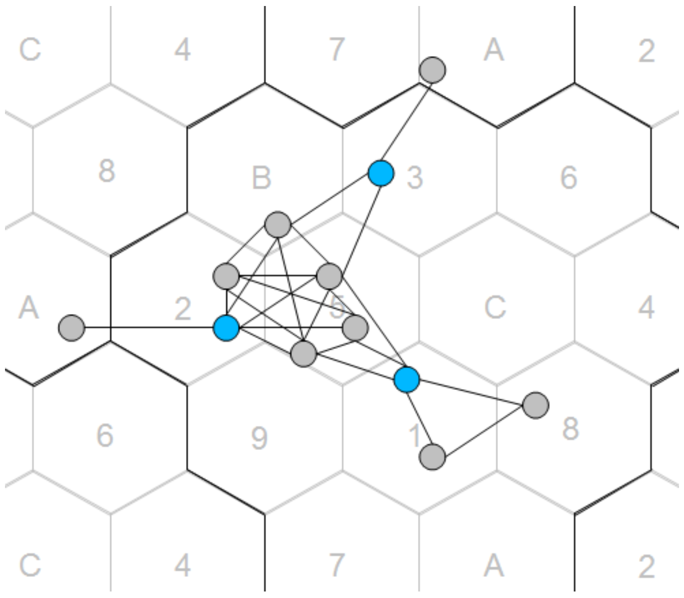


Figure 7.4: Minimal dominating set

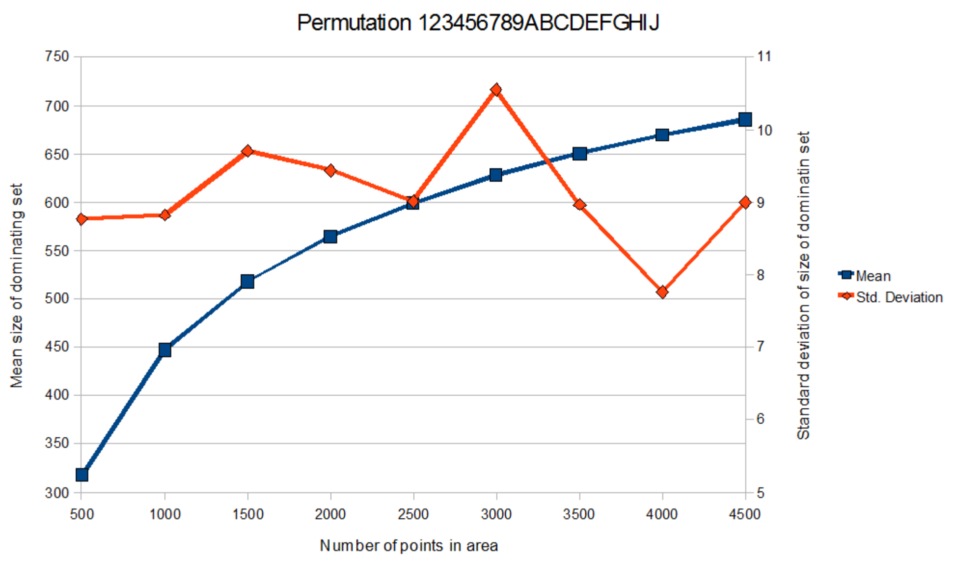


Figure 7.5: Graph for Improvement 2

Chapter 8

Documentation

In this chapter is presented the documentation for the programs and other files on DVD.

8.1 Contents of DVD

First folder includes compiled C++ programs for Windows (Win7) and Linux (x86 Debian). For usage see README.txt included, or the next section. Descriptions for algorithms are listed in readme file. It also includes Viewer.exe, program compiled for Windows described in chapter **??**.

Next folder is examples including few working examples just by copy onto writable drive and run. Input sets includes generated inputs used for statistical area. It is divided by algorithm and range. Copy and paste top level folders into /input/ to use them with algorithm.
Literature contains most articles listed in bibliography. Permutations includes permutations used in statistical data. It also includes large set of permutation, which contains only few that are isomorphic. It was generated using PHP script included in folder source codes. It was used to find good candidates for best permutation. Source codes folder also includes all source codes for programs compiled in examples. Presentations folder contains presentations demonstrating dominating and algorithms.
The last folder spreadsheets contains documents in open document format and office format, where are results from program runs and graphs being

stored. Most of the graphs are shown in this thesis.

8.2 Variables and constants documentation

Simulation program takes input from folder ‘Input/Folder/File‘, where ‘Folder‘ represents group of input files. Group of input files has input square area, number of access points and transmission range in common. All points are listed as their coordinates and the corresponding class. The program also loads permutations available in directory ‘Permutations/‘. Result is being outputted into folder ‘Output/‘. Following constants and variables are being used:

```
// Not assigned default state of access point
const int STATE_SEARCHING = 0;
// Is dominator, member of dominating set
const int STATE_LEADER = 1;
// Is being dominated by dominator
const int STATE_FOLLOWER = 2;

// Number of test - must be greater than number
// of files in input group
const int NUM_OF_TESTS = 500;
// Maximum number of input points in one file
const int MAX_RELATIVES = 40000;
// Maximum number of points in range of one point
const int MAX_RELATIVES2 = 2000;

// files for input file and input permutation
FILE * fh, * fh2;

int A; // Half of transmission range from input file
int N; // Number of points in input file
int MAX; // One side of input area square

int i,j,k,l,m,n,o,actual,last; // integer variables
int result_sum,leader,leader_max; // other variables
```

```
// variables for statistical measurements
float mean, deviation;

time_t timestamp; // timestamp
char name[50]; // input name 1
char input[50]; // input name 2
char inputdir[50], inputdir2[50]; // input paths

DIR *dp, *dpp, *dppp; // input directories
// input directory listings
struct dirent *ep, *epp, *eppp;
char perm[20]; // permutation
// char for transforming permutation
// (A = 10, B = 11, ...)
char c;

// for coverage measurements
float area,segments,distanc,seq;

// collecting results about DS
int result2[MAX_RELATIVES];
for (k=0; k < MAX_RELATIVES; k++)
{
        result2[k]=0; // inicialization
}

// collecting results about DS
int result[NUM_OF_TESTS];

// Relatives - array that represents,
// which 2 access poinst can
// communicate between each other.
// Index 0 is keeping count of relatives
// other fields are being populated
// with indexes from table sur.
int** rel = new int*[MAX_RELATIVES+1];
        for (int i = 0 ; i < MAX_RELATIVES+1 ; i++)
          rel[i] = new int[MAX_RELATIVES2+1];
```

```
// array with coordinates
// for each point, there are stored:
// 0 - x coordinate, 1 - y coordinate,
// 2 - class, 3 - state
int sur[N][4];
```

Chapter 9

Results

After this various techniques for choosing the dominator inside the cluster were examined. With study of worst case, the algorithm modification was presented to increase its effectiveness by denying the worst case possibility. This improvement led to even better effectiveness, as figure 9.1 shows.
The worst case discussed was however on relatively small density networks

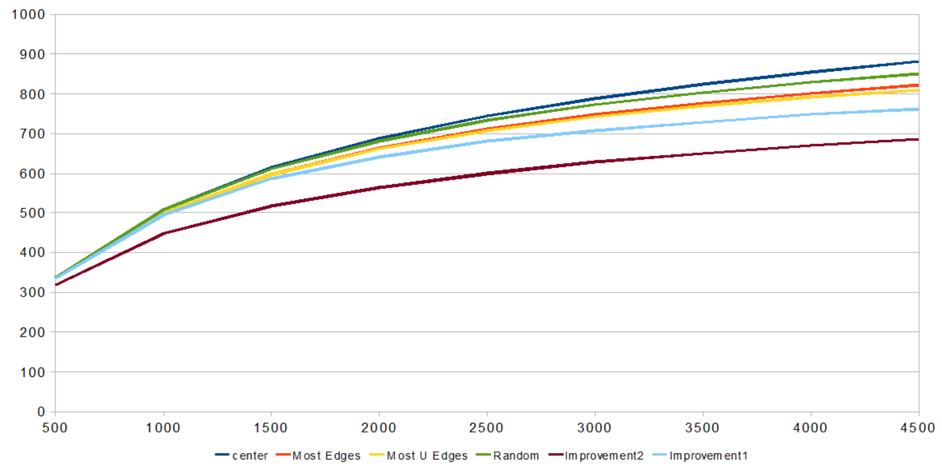


Figure 9.1: Graph showing size of dominating set for small density networks using different strategies

(average vertex degree 2-6). Therefore it is right to question, how effective are our algorithms for higher density networks. Following figure was created for higher density networks (average vertex degree 7-48).

With increasing density best algorithm is being switched from Improved 2 to

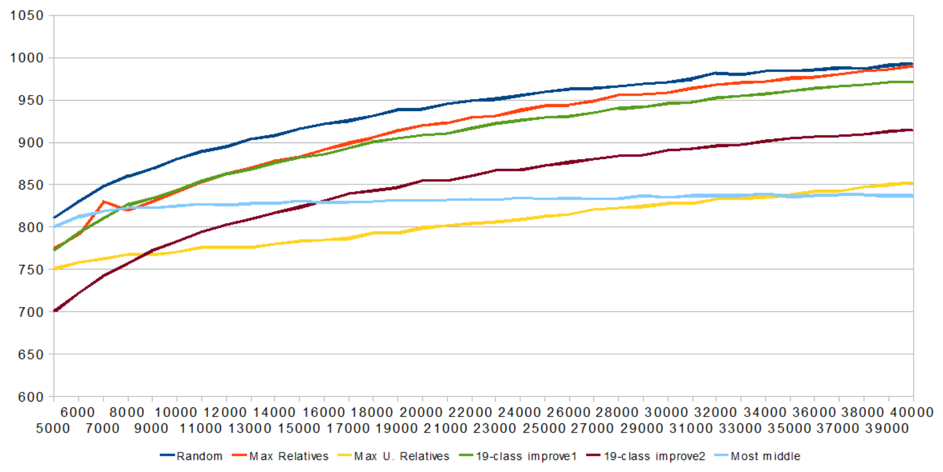


Figure 9.2: Graph showing size of dominating set for high density networks using differnet strategies

Most unassigned neighbours. It is logical, because in high density networks, there is not the same worst case, which Improved 2 algorithm is trying to solve. Finally in the end, for highest density networks the algorithm Most in center is covering most efficiently the area, therefore being best for highest density networks. This result was also confirmed by simulating with 100 000 nodes in area, where the mean size of dominating set using Most in center was 843,140015 compared to Most unassigned neighbours with 940,419983.

Summary

First of all, the best permutations have been found for this algorithm. Using these permutations the dominating set is approximately up to 12% smaller compared to using other permutations. Therefore always use best permutation, covering with first steps most of the working area. If you know your network density, you can choose most effective algorithm variant for dominating. For average vertex degree 11 and below choose algorithm Improved 2. If average vertex degree is between 11 and 41 then the best variant for dominating is Most unassigned neighbours. For higher density networks, use Most in center. Using these algorithms will get you very good approximation of minimal dominating set and also "time complexity and approximation bounds are completely independent on the size of the network." [CDF+08]
Similar algorithm working on 3 dimensional Euclidean space is presented in [AFO]. It would be interesting to know, if these variants and improvements can be also applied to this algorithm.

Bibliography

[AFO] A. E. Abdallah, T. Fevens, and J. Opatrny. 3d local algorithm for dominating sets of unit disk graphs.

[AjWF02] Khaled M. Alzoubi, Peng jun Wan, and Ophir Frieder. Distributed heuristics for connected dominating sets in wireless ad hoc networks. *Journal of Communications and Networks*, 4:22–29, 2002.

[CDF$^+$08] J. Czyzowicz, S. Dobrev, T. Fevens, H. González-Aguilar, E. Kranakis, J. Opatrny, and J. Urrutia. Local algorithms for dominating and connected dominating sets of unit disk graphs with location aware nodes. In *Proceedings of the 8th Latin American conference on Theoretical informatics*, LATIN'08, pages 158–169, Berlin, Heidelberg, 2008. Springer-Verlag.

[CED06] Deniz Cokuslu, Kayhan Erciyes, and Orhan Dagdeviren. A dominating set based clustering algorithm for mobile ad hoc networks. In *in Proc. ICCS2006, LNCS 3991*, pages 571–578. Springer-Verlag, 2006.

[DMP$^+$03] Devdatt Dubhashi, Alessandro Mei, Alessandro Panconesi, Jaikumar Radhakrishnan, and Arvind Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 717–724, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[Joh73] David S. Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, STOC '73, pages 38–49, New York, NY, USA, 1973. ACM.

[LW10] Christoph Lenzen and Roger Wattenhofer. Minimum dominating set approximation in graphs of bounded arboricity. In *Proceedings of the 24th international conference on Distributed computing*, DISC'10, pages 510–524, Berlin, Heidelberg, 2010. Springer-Verlag.

[NH04] T. Nieberg and J.L. Hurink. A ptas for the minimum dominating set problem in unit disk graphs, 2004. Imported from MEMORANDA.

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 475–484, New York, NY, USA, 1997. ACM.

[WDY08] Jie Wu, Fei Dai, and Shuhui Yang. Iterative local solutions for connected dominating set in ad hoc wireless networks. *IEEE Trans. Comput.*, 57:702–715, May 2008.

[WK09] Andreas Wiese and Evangelos Kranakis. Approximation and online algorithms. chapter Local PTAS for Dominating and Connected Dominating Set in Location Aware Unit Disk Graphs, pages 227–240. Springer-Verlag, Berlin, Heidelberg, 2009.