EFFICIENT CONSTANT K-DOMINATING SET APPROXIMATION

SEMESTER THESIS

January 27, 2005

Author
Patrice Müller
patrice@student.ethz.ch

Supervisor
Thomas Moscibroda
moscitho@tik.ee.ethz.ch

Professor
Roger Wattenhofer
wattenhofer@tik.ee.ethz.ch

Abstract

Calculating a minimal dominating set is known to be NP-complete. It would be nice to have an approximation which is not much bigger but could be calculated much faster. In this paper an algorithm is proposed which constructs a k-dominating set. It is mathematically proved that the algorithm produces a constant approximation to the MDS and terminates very quickly.

1 Introduction

In wireless ad-hoc networks collaboration among nodes is an extremely important task. In small working environments with only a few nodes in the same transmission range very simple and primitive algorithms like flooding can be used. As the networks get bigger and more complex these basic approaches to solving problems cannot be used anymore.

There needs to be established an overlay network layer on top of TCP/IP. Without it scalability and reliability cannot be guaranteed. This paper is focusing on dominating sets in unit disk graphs. There is already a lot of work available dealing with dominating set problems. However a reliable network might need to have more than just one dominator per node. In this paper it is shown how a k-dominating set with a constant approximation[i] can be constructed using only local information.

2 Algorithm

The first algorithm presented below about constructing a dominating set is taken from [1]. To extend the achieved result from dominating set to k-dominating set the algorithm was slightly modified.

2.1 Dominating Set

At the start-up of a node it needs to assign itself a random number which will be its unique identifier UID in practice if the randomization is using a large enough range of numbers. Starting with a small transmission range and in the state of a dominating set every node queries all nodes in range about their UID and selects the node with the highest UID as its dominator. Selecting here means that the node informs the selected node about its selection. The node receiving this information must accept to become a dominating node if it is not already in the dominating set. After such a dominator being found and informed the selecting node can remove its dominating state. In the special case where the selecting node is also the node with the highest UID then it will just remain in the dominating state.

For establishing a smaller dominating set the above described selection process is applied recursively to all nodes still in the dominating state. The transmission range is doubled after every round and is of the form $\delta_i = 2^i / \log n$[ii], for $i > 0$. Initially every node is in the dominating set and may fall out of it at any round. Thereafter the algorithm stops on that node. Obviously certain nodes never drop out of the dominating set or there would be no dominating set at all. After $\log\log n - 1$ rounds the algorithm must stop and

---

[i] With constant approximation an approximation to the minimal dominating set is meant. This approximation is worse only by a constant factor to the MDS.

[ii] n is the total number of nodes in the maximum transmission range of the node executing the algorithm.

the node will remain in the dominating set if no node with higher UID was found. At the $\log\log n-1^{th}$ round the transmission range is set to $\delta_{\log\log n-1}=1/2$. After this round no other node can cover all the nodes the processing node was responsible for. Even the full transmission range cannot be enough anymore to cover all non-dominating nodes.

2.2 K-Dominating Set

The same idea as above for building up a dominating set is used to construct a k-dominating set. Every node chooses a random number as its UID and starts with a small transmission range to find the k highest nodes. Those k nodes are selected and must become dominators after getting notified. If the node executing the algorithm itself is among the k highest nodes then it will remain in the dominating set. Otherwise it can remove its dominating state. If less than k nodes are available in the current transmission range then all available nodes will be notified to become a dominator as this will be the best possible approximation to a k-dominating set.
The same hierarchical technique as above for the dominating set algorithm is then applied on the remaining dominators. The transmission range will be doubled every round until the node is no longer a dominator or reaching $\delta_{\log\log n-1}=1/2$ as half of the maximum transmission range and then the algorithm stops. At each step the k highest nodes will be forced to the dominating set and the executing node will become a non-dominating node if it is not among the k highest nodes.

3 Related Work

Many ideas for this paper's work come from [1]. The following three subcategories were analyzed very carefully and then modified and extended to proof the requested properties for the k-dominating case. In all the analysis squares are supposed to be the visible range of the wireless devices. This is a method for simplifying the analysis. It does not exactly match the actual transmission range which would rather be an irregular circle depending on the nearby environment. The squares could as well be replaced by circles and the outcome of all the results would not be affected up to a constant factor.

3.1 Expectation

To get the expected number of dominators after one step of the dominating set algorithm the close proximity of one visible range is analyzed. In Figure 4.1 the model for the visible range in two dimensions is shown. It is assumed that there are no more than *m* nodes in the visible range *L* of *S* and that the side length of *S* is one half of the transmission range.
Having this setup it was shown in [1] that the expected number of dominators inside *S* is bound by $O(\sqrt{m})$. This is a very important property as it shows a drastic reduction in dominators by executing the dominating set algorithm to all the nodes in *L*.

3.2 High probability

It was also shown that the reduction of nodes inside *S* holds with high probability. To be more precise the probability that *S* contains more than $8\sqrt{m}\ln m+1$ dominators is bounded by $O(1/m^{\ln m})$. This means that there are more than $O(\sqrt{m}\ln m)$ dominators in *S* with a very small probability that goes towards zero quickly.
By knowing this the proposed algorithm guarantees a reduction in the number of dominators. Furthermore the calculated expectation is more trustful by knowing the result with high probability.

3.3 Hierarchical analysis

In the paper [1] it is said that the recursive algorithm will eventually produce a constant approximation $O(d)$ to the minimal dominating set with *d* dominators by using at most $\log\log n-1$ steps. To show this property the expectation of $O(\sqrt{m})$ for one step was iterated on itself. This calculation cannot be regarded as correct as the expected reduction of the remaining dominators may change after one round of execution. The first round is not and cannot be considered to be independent from the following rounds. For analyzing the hierarchical algorithm and showing a constant approximation of the minimal dominating set a different approach needs to be used.

4 K-dominating set

Constructing a dominating set can be a very important task in wireless ad-hoc networks. Generally there is no structure among wireless devices as they can be located anywhere. In certain areas the density may be very high and in other parts it may be very sparse. Building up a dominating set can help to solve problems like routing. The standard dominating set has the property that every node has at least one node in its range as its dominator. If we consider using such a dominating set for instance for solving a routing problem it is easy to see that the whole system is not very fault tolerant. If the dominating set problem can be extended to be more reliable even if nodes are unstable then problems may be solved on top of this k-dominating environment by ensuring a higher reliability.
The algorithm for constructing such a k-dominating set is already given above. In this section it is to prove its properties. The first step is to show that the algorithm reduces the amount of dominators. This is done by analyzing the expected number of dominators after the execution of one step of the algorithm.
In a second analysis it is shown that not only the expectation holds but that a slightly higher result holds with high probability. This proof is essential to be convinced that the algorithm almost cannot be unfavorable.
In a third subchapter the whole algorithm using its hierarchical structure is analyzed. This leads to the final result and does show that the proposed algorithm fulfills the wanted requirements.

4.1 Expectation

At a first step to analyze the algorithm for k-dominating sets the reduction in dominators is calculated. To do this one round of the algorithm is taken a close look at. In Figure 4.1 the model used can be seen. The expectation for the number of dominators inside *S* after one round is calculated. All the areas are squares to simplify the setup. The side lengths are equal to 1/2. There is a central area *S* and a neighboring area *S'*. In total there are eight equal neighboring areas to *S*. Therefore only one neighboring area needs to be analyzed in relation to the central area. The other seven neighboring areas do have the exact same properties.

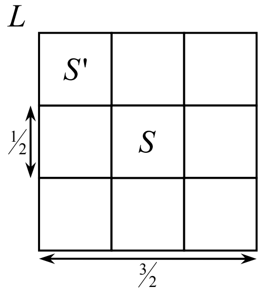

Figure 4.1: Visible range L of S

First the number of nodes nominated from inside S must be examined. The side length of *S* is assumed to be half the transmission range of the nodes executing the algorithm. Due to this fact all the nodes residing inside *S* are mutually visible. Every node knows about all the other nodes' presence and also knows all their UIDs. The algorithm nominates from every node k nodes to become their dominators. As a result of the mutual visibility no more than k nodes inside *S* can become dominators from nodes within *S*.

(4.1) $$E_1 \le k$$

Second the number of nodes nominated from outside *S* must be examined. Let x be the number of nodes in *S* and y inside *S'*.

$$x = |S| \ , y = |S'|$$

Any node $p \in S$ can be nominated by a $q \in S'$ only if *p* has its UID within the k-largest of *q's* visibility. The random selection of UIDs at the beginning of the algorithm gives a uniformly distribution of UIDs on *S* and *S'*. Therefore the probability for node p being within the k-highest nodes of *q's* visibility is

(4.2) $$P \le \frac{k}{1+y}.$$

Having *x* nodes in *S* the expected number of nodes nominated in *S* from *S'* is

(4.3) $$E_{21} \le \frac{xk}{1+y}.$$

Those points are nominated to be dominators from points in *S'*. As there are *y* points in *S'* there is an absolute maximum of nominated points inside *S* from points $q \in S'$ of

(4.4) $$E_{22} \le yk.$$

With the two expectations E_{21} and E_{22} the overall expectation for nominated points from *S'* can be calculated. The variable *m* is the total number of nodes in *L* before the algorithm starts.

(4.5) $$\begin{aligned} E_2 &= \min\left(\frac{xk}{1+y}, yk\right) \\ &\underset{\text{as } x,y \ge 0}{=} k \min\left(\frac{x}{1+y}, y\right) \\ &\le k\left(\sqrt{x+y+1}-1\right) \\ &< k\sqrt{m} \end{aligned}$$

To get the total expectation the nodes nominated from inside and those nominated from outside of *S* must be added. There are 8 equal squares around *S*. A new constant *c* is used to simplify the equation.

(4.6) $$\begin{aligned} E &= E_1 + 8E_2 \\ &< k + 8k\sqrt{m} \\ &< c\sqrt{m} \end{aligned}$$

From this result the upper bound of expected dominators after one round can be seen easily. It is

(4.7) $$E \le O\left(\sqrt{m}\right).$$

4.2 High probability

In this subchapter a stronger bound than the expectation is shown. By knowing only the expectation the distribution of all the possible cases remains unknown. It would be nice if almost all the possibilities were close to the expectation. In this case the expectation seems to be a very reasonable bound. But what if the distribution is not so favorable? There might be a very high probability for almost no reduction in dominators and a high probability of a very good reduction. If the algorithm happens to produce the unfavorable reduction it doesn't help much to know about the expectation since a good reduction is aimed at. Showing a good reduction with high probability is much more convincing that the proposed algorithm works with all circumstances and at all times.

As a first step to get a high probability bound the probability that the number of nodes nominated in S from *S'* can reach a certain benchmark s is calculated. Once this function is evaluated a benchmark can be searched. This benchmark will be somewhere above the number in expectation and must lead to a very low probability.

The probability that s (or more) points in *S* can be nominated from *S'* is needed. To get this probability the two areas *S* and *S'* are analyzed in detail.

Lemma 4.1: Out of the s+k-1 highest nodes in $S \cup S'$ at least s must be in *S* for *S'* being able to nominate s or more dominators in *S*.

Proof: If there are less than s of the s+k-1 highest nodes in *S* than the nodes in *S'* have no possibility to nominate s dominators in *S*. They are forced to select more nodes in *S'* as their dominators. If the s nodes in *S* are chosen from a smaller group than the s+k-1 highest nodes not all the possibilities for having selected s dominators in S are covered. This comes from the fact that each node in *S'* chooses the k highest nodes as dominators. Therefore it is possible that a node chooses the first k-1 dominators in *S'* and the kth dominator in *S*. For this reason the s+k-1 highest group must be considered. □

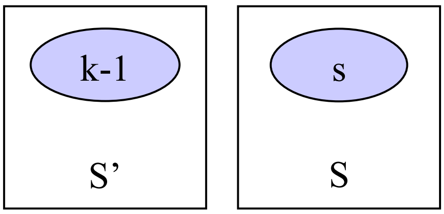

Figure 4.2: Distributing the s+k-1 highest nodes

Knowing Lemma 4.1 the probability can be calculated by analyzing the combinations. The number of possibilities for choosing the s points in S is

$$\binom{x}{s}. \tag{4.8}$$

The number of possibilities for choosing the k-1 points in S' is

$$\binom{y}{k-1}. \tag{4.9}$$

The number of possibilities for choosing the s+k-1 highest points among the nodes $S \cup S'$ (containing x,y nodes respectively) is

$$\binom{x+y}{s+k-1}. \tag{4.10}$$

At this point the probability can be written by using

$$\frac{\#\text{number of 'good' cases}}{\#\text{number of total cases}}$$

which leads to the equation

$$p = \frac{\sum_{i=s}^{s+k-1} \binom{x}{i}\binom{y}{s+k-1-i}}{\binom{x+y}{s+k-1}}. \tag{4.11}$$

In the above equation the top part adds up all the possibilities for having more than s nodes in S out of the s+k-1 highest nodes. The lower part counts all the possibilities for placing the s+k-1 highest nodes among the x+y nodes of $S \cup S'$.

By testing the probability of equation (4.11) a fast degradation can be observed. Using the same asymptotical bound as in [1] for the proof with high probability then also the same low probability is found. It was proofed by Thomas Moscibroda that with an exponentially small probability

$$p \le \frac{1}{m^{\log m}} \tag{4.12}$$

a maximal bound of

$$O\left(\max\left(\sqrt{m}\log m, 2k\right)\right) \tag{4.13}$$

exists as m gets large. By knowing this result the proposed algorithm for the k-dominating set seems to be very assuring to decrease the number of dominators inside S significantly with one step. When the number of nodes is high the resulting number of dominators will be $< \sqrt{m}\log m$ with very high probability. The number of dominators can always reach $2k$ which is just a constant number and can be assumed to be small.

Now an upper bound with high probability was shown. This bound is just a little higher than the value in expectation. Due to this fact the algorithm performs well in all practical cases.

4.3 Hierarchical Analysis

In this section the whole k-dominating set algorithm is analyzed. In the previous sections only one step of the algorithm was executed. The whole process continues though by doubling the transmission range after each round and proceeding with the remaining dominators. In [1] only the expectation of dominators was recursively calculated to show a constant approximation at the end. This calculation is misleading as consecutive rounds are not fully independent.

For the mathematical analysis in this section the high probability result is used. All the possibilities combined with their number of remaining dominators are added up. This will give the final expectation of dominators under a high probability condition after the algorithm terminates.

The reduction in dominators from n down to a constant approximation is divided into 2 phases. In Figure 4.3 it can be seen that the first phase reduces the dominators to a $\log\log n$ approximation and the second phase brings the algorithm to its stop at a constant approximation.

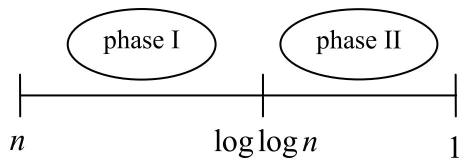

Figure 4.3 Hierarchical algorithm divided into 2 phases for mathematical analysis

4.3.1 Phase I $(n \to \log\log n)$

Starting from a total number of n dominators[i] in the whole environment[ii] the number gets reduced at each step with a high probability. Still there remains a slight possibility that the number doesn't get reduced very much. At the beginning there are $n_0 = n$ dominators and after the i^{th} round there remain n_i dominators. For round $i+1$ only the 'good' $(+)$ cases up to round i are of interest. As soon as a 'bad' $(-)$ case happens the execution stops for the analysis and counts the number of dominators before the 'bad' case. Following the first 3 rounds are listed.

Round 1

$(-)$ $\quad p = \frac{1}{n_0^{\log n_o}}$

$$n_1 \le n_o$$

[i] Remember that at the beginning of the k-dominating set algorithm all nodes are marked to be in the dominating set and remain dominators until they are not nominated anymore at some round.

[ii] At this stage the interest of the analysis is on all nodes in a square with a side length of half the maximal transmission range and not only on same specific area S anymore. Only this analysis will lead to the desired constant overall approximation.

(+) $p = 1 - \frac{1}{n_0^{\log n_o}}$

$n_1 \le \sqrt{n_o} \log n_o$

Round 2

(−) see Round 1

(+−) $p = \left(1 - \frac{1}{n_0^{\log n_o}}\right)\left(\frac{1}{n_1^{\log n_1}}\right)$

$n_2 \le \sqrt{n_o} \log n_o$

(++) $p = \left(1 - \frac{1}{n_0^{\log n_o}}\right)\left(1 - \frac{1}{n_1^{\log n_1}}\right)$

$n_2 \le \sqrt{n_1} \log n_1$

Round 3

(−) see Round 1

(+−) see Round 2

(++−) $p = \left(1 - \frac{1}{n_0^{\log n_o}}\right)\left(1 - \frac{1}{n_1^{\log n_1}}\right)\left(\frac{1}{n_2^{\log n_2}}\right)$

$n_3 \le n_2$

(+++) $p = \left(1 - \frac{1}{n_0^{\log n_o}}\right)\left(1 - \frac{1}{n_1^{\log n_1}}\right)\left(1 - \frac{1}{n_2^{\log n_2}}\right)$

$n_3 \le \sqrt{n_2} \log n_2$

From the above first 3 rounds it is now easy to deduce the expected number of dominators for the 'bad' cases in the first phase. To do this the sum of the probabilities multiplied with the number of dominators of all the cases with a 'bad' finishing round must be calculated. To simplify the calculation the probabilities are augmented a little bit. Instead of using the true and high probabilities of having the 'good' cases before the aborting 'bad' case, those probabilities are set to 1. Using this simplification the expected number of dominators from the first phase can be calculated by

(4.14) $$E[Phase1] = \sum_{i=0}^{n_i > \log\log n} \frac{n_i}{n_i^{\log n_i}} = \sum_{i=0}^{n_i > \log\log n} \underbrace{\frac{1}{n_i^{\log n_i - 1}}}_{S_i}$$

Now two properties must be show for the function (4.14) to receive a constant expectation. First it must be shown that the term S_i is sufficiently small and second it must be shown that there are not too many additions.

Lemma 4.2: $n_i > \log\log n \Rightarrow S_i < \frac{1}{\log\log n}$

Proof: For a n_i that is large enough it is true that $\log n_i - 1 > 1$. As n is not a constant $n_i > \log\log n$ can be assumed to meet the previous requirement. Out of this follows immediately that $S_i = \frac{1}{n_i^{\log n_i - 1}} < \frac{1}{n_i} < \frac{1}{\log\log n}$. □

Lemma 4.3: There are only $O(\log\log n)$ steps in applying $\sqrt{n_i}\log n_i$ recursively to itself until the result is smaller or equal to $\log\log n$.

Proof: Using Maple it was shown that the number of recursions is growing proportionally slower than $\log\log n$ for $n = 10^3, 10^6, 10^{20}$. The number of recursions with 1000 nodes is 11 and $\log\log(1000) \approx 1.93$. This means there are about 6 times more recursions as with $\log\log n$. Due to the structure of the functions analyzed they have no discontinuities and will not change their behavior in the large. Therefore the number of recursions is bound as Lemma 4.3 says. □

Now the expectation for the first phase can be calculated using function (4.14) leading to the final expectation of the first phase

(4.15) $$E[Phase1] \le \sum_{i=0}^{c\log\log n} S_i < \frac{c\log\log n}{\log\log n} \le O(1).$$

As shown for Lemma 4.3[i] $c \le 6$ is true if $n \ge 10^3$. Therefore the above result proofs that the number of dominators from phase 1 is bound by $O(1)$.

4.3.2 Phase II $(\log\log n \to 1)$

For the second phase as seen in Figure 4.3 the number of dominators must be reduced from $\log\log n$ to 1. As the analysis here is continued from the previous result the number of dominators is $\le \log\log n$.

At this stage when the number of dominators is getting fewer then the probability for 'bad' cases is getting bigger. It is not possible just to sum up all the cases as in the first phase or the expected number of dominators from the second phase will be too high. To proceed the second phase is divided into X rounds. As it would be nice to have the algorithm achieve the constant approximation within $\log\log n$ rounds the maximum number of rounds is set to

(4.16) $$X \le \log\log n.$$

Each round now has a failure rate[ii] of

(4.17) $$P[failure] = \frac{1}{m_i^{\log m_i}} < \frac{1}{2}, \quad (m_i \ge 3)$$
$$m_{i+1} = m_i$$

Out of this equation follows immediately the success rate at each round which is equals to

(4.18) $$P[success] \ge \frac{1}{2}, \quad (m_i \ge 3)$$
$$m_{i+1} = \sqrt{m_i}\log m_i$$

From here on the second phase is divided into 2 cases. The first one is that there are $\le X/4$ successful rounds on the total of X rounds. The second case is the one with $> X/4$ successful rounds. For solving the first case the Chernoff bound is very helpful.

Chernoff bound:

(4.19) $$P\left[E \le (1-\delta)\mu\right] \le e^{-\delta^2\mu}$$

Applying the Chernoff bound to the first case the probability for having $\le X/4$ successful rounds can be calculated with

[i] For the numerical analysis the logarithm to the base 10 was used.

[ii] The failure rate is equal to the probability of having a 'bad' case in that particular round.

$$(4.20)\ P\left[E \le \frac{X}{4}\right] = P\left[E \le \left(1-\frac{1}{2}\right)\frac{X}{2}\right] \le e^{-\frac{1}{4}\cdot\frac{X}{2}} = e^{-\frac{X}{8}}.$$

Using the above probability the expected number of dominators from the first case can be calculated. To simplify the calculation the worst case is taken where no reduction has taken place since reaching $\log\log n$ dominators. Therefore the expectation is

$$(4.21)\qquad E[case1] \le e^{-\frac{X}{8}} \cdot \log\log n.$$

As X was bound by the assumption in (4.16) the above equation can by simplified as follows.

$$(4.22)\qquad \begin{aligned} E[case1] &\le e^{-\frac{\log\log n}{8}} \cdot \log\log n \\ &= (\log n)^{-\frac{1}{8}} \cdot \log\log n \\ &= \frac{\log\log n}{\sqrt[8]{\log n}} \\ &\le O(1) \end{aligned}$$

Now it is also shown that the first case of the second phase bounds the number of dominators by $O(1)$.

It remains to show that the second case doesn't leave too many dominators either. Continuing with the dominators reduced to $\le \log\log n$ the algorithm executes and due to the precondition there are more than $X/4$ rounds. These 'good' rounds all lower the number of dominators to $m_{i+1} = \sqrt{m_i}\log m_i$. It was shown that the recursive application of the above implicit reduction function can be bound with an exponentially decreasing explicit function

$$(4.23)\qquad \sqrt{m}\log m \le m^{3/4}\ ,\ m \ge 5504.$$

This makes it possible to use the explicit function for estimating the upper bound of expected remaining dominators for the second case. The probability is just set to 1 as this will lead to a sufficiently small bound.

$$(4.24)\qquad \begin{aligned} E[case2] &\le 1\cdot(\log\log n)^{(3/4)^{(X/4)}} \\ &= (\log\log n)^{(3/4)^{\log\log n}} \quad \text{(I)} \\ &= (\log\log n)^{(\log n)^k}\ ,\ k \approx -0.2876 \\ &\le (\log\log n)^{(\log n)^{-1/4}} \\ &\le \sqrt[4]{\log n}^{\sqrt[4]{\log n}} \quad \text{(II)} \\ &\le O(1) \quad \text{(III)} \end{aligned}$$

In the above calculation for the second case several properties were used. These properties are listed here.

$$\text{(I)}\qquad X \le 4\log\log n$$

$$\text{(II)}\qquad \log\log n \le \sqrt[4]{\log n}$$

$$\text{(III)}\qquad X^{1/X} \le O(1)$$

With (4.24) it is show that the second case of phase 2 reduces the number of remaining dominators below the bound of $O(1)$.

4.3.3 Overall bound

Due to the split up of this hierarchical analysis the individual parts need to be combined to get the overall bound for the whole algorithm. The total process was divided into 3 different parts. In the first phase the possible dominators due to failures were estimated. In the second phase two cases had to be considered. The first case is unfavorable as it has few successful rounds. The number of possible dominators still stays small as the possibility for this unfavorable case is relatively small. The last case is the 'good' case when there are many favorable rounds. In this case it is almost obvious that the number of dominators gets reduced as desired. Summing up all the expectations the overall bound is found by

$$(4.25)\qquad E \le \underbrace{O(1)}_{4.15} + \underbrace{O(1)}_{4.22} + \underbrace{O(1)}_{4.24} = O(1).$$

5 Conclusion

In this paper it is shown how a k-dominating set can be constructed using only local information. The proposed hierarchical algorithm performs this task and it could be shown that it achieves an overall bound of $O(1)$ dominators in each unit disk square of side length 1/2. Suppose a minimal dominating set contains *d* dominators in the whole environment. The proposed hierarchical algorithm can cover the whole environment with *4d* unit squares of side length 1/2 and all of them having $O(1)$ dominators. It follows immediately that this paper's algorithm constructs a k-dominating set using only local information with an $O(d)$ approximation to the optimal dominating. The time require for execution of the algorithm can also be bound asymptotically. Both phases of the hierarchical analysis terminate in $O(\log\log n)$. Therefore the total algorithm also terminates in $O(\log\log n)$.

6 Future work

The mathematical analysis of this paper focuses mainly on the asymptotical bound. Certain functional transformations can only be realized when the number of nodes is very big. It may be necessary for future work to analyze the algorithm in detail on a 'reasonable' number of nodes depending on the required application.

Acknowledgments

For the supportive and indispensable suggestions I would like to thank my supervisor Thomas Moscibroda. Many crucial steps in the mathematical analysis would not have been possible without his help. A very grateful thank you also goes to Fabian Kuhn who helped to find the right track in several discussions.

References

[1] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, An Zhu. Discrete Mobile Centers, pages 1-11, 2003.

[2] Fei Dai, Jie Wu. An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks, pages 1-4, 2004.