Rheinische Friedrich-Wilhelms-Universität Bonn
Mathematisch-Naturwissenschaftliche Fakultät

Theoretical Aspects of Intruder Search

MA-INF 1318 Manuscript Wintersemsester 2015/2016

Elmar Langetepe

Bonn, 19. October 2015

The manuscript will be successively extended during the lecture in the Wintersemester. Hints and comments for improvements can be given to Elmar Langetepe by E-Mail `elmar.langetepe@informatik.uni-bonn.de`. Thanks in advance!

Lemma 15 *If a vertex at depth d is burning in an optimal strategy for an instance of the firefigther problem on trees, at least $\frac{1}{2}(d^2+d)$ vertices are safe.*

Proof. Let us assume that in T and for an optimal startegy, a vertex v at depth d is burning. Then by Lemma 12 there is a protected vertex v_i for any depth $i = 1, \ldots, d$. Any tree T_{v_i} should contain at least $d - i + 1$ vertices. Otherwise it was better to choose a vertex along the path from r to v in this step i. Thus

$$\sum_{i=1}^{d}(d-i+1) = \frac{1}{2}(d^2+d)$$

gives the bound. □

Theorem 16 *There is an $O\left(2^{\sqrt{2n}}n^{3/2}\right)$ algorithm for the firefigther problem on a tree of size n.*

Proof. We show that we can run the algorithm of Theorem 13 for $k \leq \sqrt{2n}$. Suppose a vertex of depth $\sqrt{2n}$ is burning. Then by Lemma 15 $n + \sqrt{n/2} > n$ vertices are safe witch contradicts the number n of vertices. This means that all vertices of depth $\sqrt{2n}$ are safe in an optimal strategy. In turn an optimal strategy makes use of less than $\sqrt{2n}$ guards. Thus we set $k \leq \sqrt{2n}$ which gives the bound. □

2.2.3 Capture of an Intruder by moving agents

Up to now we have considered stationary guards that check and block vertices of the graph or tree. Many other variants are known from the literature. We discuss the following version. A set of k agents starts at a homebase vertex b in a given tree $T = (V, E)$ and there is some intruder somewhere in T. The following problem definition and its solution goes back to Barrière et al. [].

The intention is to *clear* all *edges* of the tree. A *contiguous search strategy* can perform one of the following two operations in one search time step:

1. Place a team of p guards on a vertex.
2. Move a team of r guards along an edge.

Additionally, the set of all *cleared* edges E_i after step i has to be connected for any i.

In one time step a set of agents located at vertex u can move along an edge $e = (v, u)$ from v to u. There is an integer edge weight $w(e) \geq 1$ that says how many agents have to move along e so that the intruder cannot cross the edge from u to v in this step. In this sense the agents *clear* the link in this case. Intuitively, we can think of a large corridor and cleaning the corridor from v to u requires at least $w(e)$ agents. Furthermore, we have integer weights $w(v)$ for the vertices. Locally, a clear link $e = (v, u)$ is preserved from recontamination, if either the vertices v and u are guarded by $w(v)$ and $w(u)$ agents, respectively or all other edges incident to e are also clear.

We assume that guarding a vertex is at least as hard as the maximum number of searchers required for clearing an incident edge. Thus, we require $w(v) \geq w(e)$ for any $e = (v, u) \in E$. So for any vertex $v \in V$ the weight could for example be given by $w(v) := \max_{e=(u,v)\in E} w(e)$. An example for a weighted tree is given in Figure 2.5.

In any time step (after a step of the search strategy) an edge e becomes recontaminated, if there is a path from a contaminated edge e' to edge e that is not blocked by agents on the vertices.

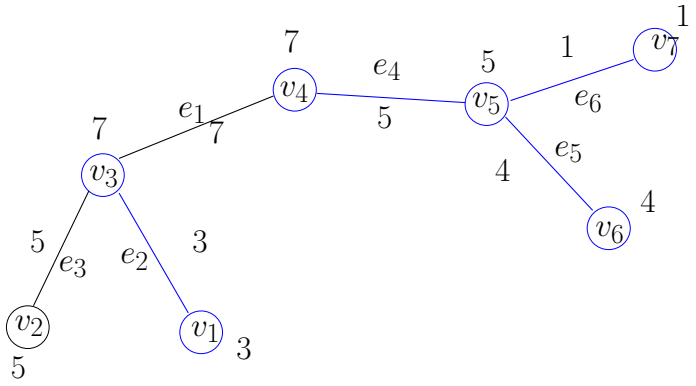

Figure 2.5: A given weighted tree $T = (V, E)$. A successful contiguous strategy can start with 10 agents at v_1, also $\mathrm{cs}(T) = 10$ holds. The size of the frontiers of $X_1 = \{e_4, e_5, e_6\}$ and $X_2 = \{e_2\}$ is $w(X_1) = 7$ and $w(X_2) = 10$, respectively.

This means that a corresponding intruder has infinite speed for crossing an arbitrary number of edges. We consider contiguous search strategies as described above and ask for the minimum number of guards, $\mathrm{cs}(T)$, that suffices to finally clear all links of the tree T.

A contiguous strategy for Figure 2.5 can start with 10 agents in vertex v_7, clears edge e_6. Leaves 6 agents at v_5 and clear e_5 with 4 agents. Moves back with 4 agents to v_5 and clears edge e_4 and edge e_7 with 10 agents. Now, we leave 7 agents at v_3, clear e_2 by 3 agents, move back with three agents and finally clear the last edge e_3. Note that in the second last step, if we leave less than 7 agents at v_3, the edge e_3 is contaminated and can recontaminate the full tree.

We would like to consider such monotone strategies. That is, a link that has already been cleaned at some step t should not be contaminated (or visited by the intruder) later again. This means that some of the agents will become stationary for a period of time in order to block the movement of the intruder (or the recontamination of links). Additionally, we can compute optimal monotone strategies efficiently as we will see below. First, we show that monotone strategies always exist.

2.2.4 Existance of monotone strategies

One main structural result is, that for a tree T for the computation of $\mathrm{cs}(T)$ it suffices to consider a montone strategy, where all agents start at the same homebase vertex b.

Theorem 17 *For any weighted tree T there is a monotone contiguous search strategy with $cs(T)$ agents where all agents initially start at the same vertex b.*

For the proof we require some notations. Let $T = (V, E)$. For a subset $X \subseteq E$ we denote all vertices that have a vertex incident to X and $E \setminus X$ as the *boundary vertices* $\delta(X)$. If X_i denotes the set of clear links after time step i of a strategy then at least

$$w(X_i) := \sum_{v \in \delta(X_i)} w(v) \tag{2.1}$$

guards have been used. (Note that the contamination threatens not only the directly adjacent links but also any fully non-protected path!)

In Figure 2.5 we have $w(\{e_4, e_5, e_6\}) = 7$ and $w(\{e_2\}) = 10$.

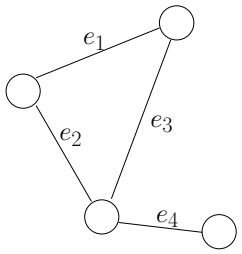

Figure 2.6: Crusades are only defined by subsets of edges of a graph. For the given graph G there is a connected crusade $(\emptyset, \{e_1\}, \{e_1, e_2\}, \{e_2\}, \{e_2, e_3\}, \{e_1, e_2, e_3\}, \{e_3, e_4\}, \{e_1, e_3, e_4\}, \{e_1, e_2, e_3, e_4\})$ and a pogressive connected crusade is $(\emptyset, \{e_1\}, \{e_1, e_2\}, \{e_1, e_2, e_3\}, \{e_1, e_2, e_3, e_4\})$.

For $G = (V, E)$ a sequence $(X_0, X_1, \ldots, X_m)$ of subsets $X_i \subseteq E$ is called a *crusade*, if $X_0 = \emptyset$ and $X_m = E$ and $|X_i \setminus X_{i-1}| \leq 1$ for $1 \leq i \leq m$. The *frontier* of a crusade $(X_0, X_1, \ldots, X_m)$ is given by $\max_{1 \leq i \leq m} w(X_i)$ as defined in Equation 2.1 above, which is the minimum number of guards required for defending any X_i. A crusade is *progressive* if $X_0 \subseteq X_1 \subseteq \cdots \subseteq X_m$ and $|X_i \setminus X_{i-1}| = 1$ for $1 \leq i \leq m$. The crusade is *connected*, if X_i is connected for $1 \leq i \leq m$.

Note that the definition of crusades holds for arbitrary graphs G. See Figure 2.6 for an example.

There will be a progressive connected crusade that will finally describe a strategy. In the very beginning it is allowed to set agents on the vertices in many non-crusading steps. Then the crusade starts to clean the links. But keep in mind that the above (connected) crusades are only defined over sets of links up to now and do not directly represent a strategy.

Exercise 11 *Define a crusade $C = (X_0, X_1, \ldots, X_m)$ of a tree T that is not related to a strategy.*

It is easy to see, that for $\mathrm{cs}(T) \leq k$ there should be a connected crusade of frontier $\leq k$ in T. For a given contiguous strategy S let $C = (X_0, X_1, \ldots, X_m)$ denote the sequence of clear links after each search step i. In any search step we can clear at most one additional edge, which means $|X_i \setminus X_{i-1}| \leq 1$ for $1 \leq i \leq m$. Since X_i is connected (definition of contiguous) and is no further destructed after search step i, we have $w(X_i) \leq k$. Of course by construction we have $X_0 = \emptyset$ and $X_m = E$. An example for Figure 2.5 is the above mentioned strategy and the connected crusade $C = (\emptyset, \{e_6\}, \{e_6, e_5\}, \{e_6, e_5, e_4\}, \{e_6, e_5, e_4, e_1\}, \{e_6, e_5, e_4, e_1, e_2\}, \{e_6, e_5, e_4, e_1, e_2, e_3\})$

We conclude:

Lemma 18 *For $cs(T) \leq k$ there is a connected crusade of frontier at most k.*

Now we would like to show that there is also a progressive connected crusade in T of frontier at most k. Note that the above connected crusade of Figure 2.5 is indeed progressive.

So starting from the above Lemma from all connected crusades $C = (X_0, X_1, \ldots, X_m)$ of frontier at most k we choose one that satisfies the following properties:

1. $\sum_{i=0}^{m} (w(X_i) + 1)$ is minimum.
2. Amog all crusade satisfying condition 1. choose one with: $\sum_{i=0}^{m} |X_i|$ is minimum.

Obviously, one such crusade has to exist. So we only have to show that the crusade is progressive. This means that we have to show $X_0 \subseteq X_1 \subseteq \cdots \subseteq X_m$ and $|X_i \setminus X_{i-1}| = 1$ for $1 \leq i \leq m$.

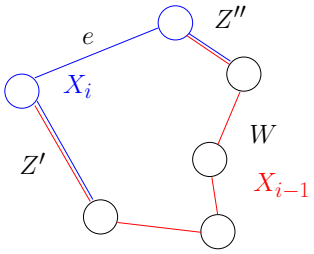

Figure 2.7: Let $\{e\} = X_{i-1} \setminus X_i$ and $W = X_i \setminus X_{i-1}$ and $Z = X_{i-1} \cap X_i$. By assuption $Z = Z' \cup Z''$ where Z' and Z'' do not share a vertex. Thus, the cycle $Z' \cup Z'' \cup W \cup \{e\}$ contradicts T.

Let us first assume that $|X_i \setminus X_{i-1}| = 0$ holds, which means that $X_i \subseteq X_{i-1}$. Therefore, we consider the connected crusade

$$C' = (X_0, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m)$$

of frontier at most k which contradicts condition 1. Note that $|X_{i+1} \setminus X_{i-1}| \leq 1$ can be concluded from $|X_{i+1} \setminus X_i| \leq 1$ and $X_i \subseteq X_{i-1}$.

Thus, we can assume that $|X_i \setminus X_{i-1}| = 1$ holds for $1 \leq i \leq m$. Next, we prove $X_{i-1} \subseteq X_i$ for $1 \leq i \leq m$. We will first conclude that

$$w(X_{i-1} \cup X_i) \geq w(X_i) \tag{2.2}$$

holds for $1 \leq i \leq m$. Otherwise, we make use of the crusade

$$C' = (X_0, \ldots, X_{i-1}, X_{i-1} \cup X_i, X_{i+1}, \ldots, X_m) \tag{2.3}$$

of frontier at most k which is a contradiction to condition 1. We know that X_i and X_{i-1} are connected. Thus, the set $X_{i-1} \cup X_i$ is connected since $|X_i \setminus X_{i-1}| = 1$ holds. We can also conclude that $|X_{i+1} \setminus (X_{i-1} \cup X_i)| \leq 1$ holds, since $|X_{i+1} \setminus X_i| = 1$. If $|X_{i+1} \setminus (X_{i-1} \cup X_i)| = 0$ holds we can go back to the first step again.

Now assume that Equation 2.2 holds. For any two arbitrary link sets A and B of T we have $w(A \cup B) + w(A \cap B) \leq w(A) + w(B)$ which is the question of Exercise 12. We conclude $w(X_{i-1} \cap X_i) \leq w(X_i)$ for $1 \leq i \leq m$ from Equation 2.2 and consider

$$C'' = (X_0, \ldots, X_{i-2}, X_{i-1} \cap X_i, X_{i+1}, \ldots, X_m)\,. \tag{2.4}$$

Now, C'' has frontier at most k.

By the minimality of C w.r.t. condition 2. we conclude that $|X_{i-1} \cap X_i| \geq |X_{i-1}|$ and therefore $X_{i-1} \subseteq X_i$ holds. We conclude $|X_i \setminus (X_i \cap X_{i-1})| = |X_i \setminus X_{i-1}| = 1$ and $|(X_i \cap X_{i-1}) \setminus X_{i-2}| \leq |X_{i-1} \setminus X_{i-2}| \leq 1$.

It remains to show that $X_{i-1} \cap X_i$ is also connected. Assume that this is not the case. Let $\{e\} = X_i \setminus X_{i-1}$ and $W = X_{i-1} \setminus X_i$ and $Z = X_{i-1} \cap X_i$. By assumption $Z = Z' \cup Z''$ where Z' and Z'' do not share a vertex. The situation is depicted in Figure 2.7. We have $X_{i-1} = Z' \cup Z'' \cup W$ and $X_i = Z' \cup Z'' \cup \{e\}$. Since X_i and X_{i-1} are both connected, and $\{e\} \notin W$ there is a cycle $Z' \cup Z'' \cup W \cup \{e\}$ in T which contradicts to the assumption that T is a tree, $X_{i-1} \cap X_i$ is also connected. We have proven the following Lemma.

Lemma 19 *For $cs(T) \leq k$ there is a progressive connected crusade of frontier at most k in T.*

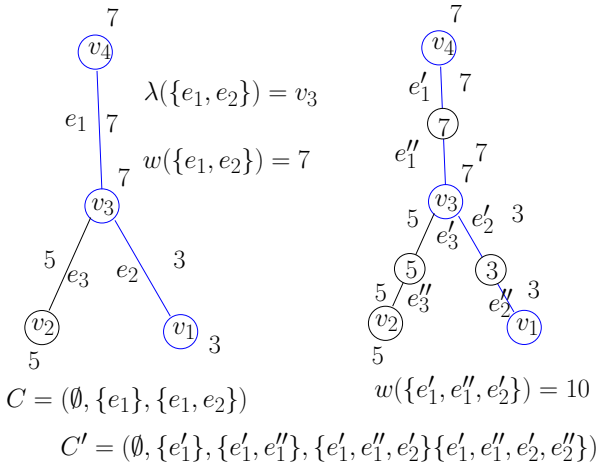

Figure 2.8: Replacing any link e in the tree T by two consecutive links e' and e'' yields the desired correspondance of the frontier and the frontier and the number of agents required for $C = (\emptyset, \{e_1\}, \{e_1, e_2\})$.

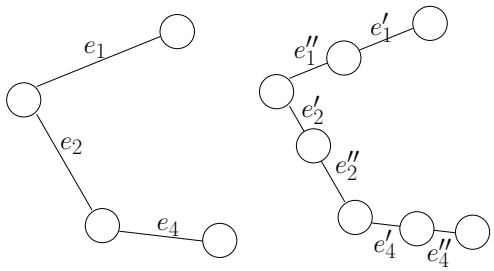

Figure 2.9: Replacing any link e in the tree T by two consecutive links e' and e''. Any link in T' has at least one vertex of degree 2.

Exercise 12 *For two link (edge) sets A and B in a graph G prove that $w(A \cup B) + w(A \cap B) \leq w(A) + w(B)$ holds.*

Now, for obtaining a monotone contiguous strategy from above progressive connected crusade, we first extend the tree T. We replace every link e by two consecutive links e' and e'' of the same weight $w(e)$. The newly inserted vertex v has also weight $w(v) := w(e)$. This means that after this transformation any link has at least one vertex of degree 2. Let T' be the outcome of this process. We will use this property for designing a monotone contiguous strategy from a progressive connected crusade. Afterwards we can transform the strategy back for acting on T. Obviously, also $\mathrm{cs}(T') \leq \mathrm{cs}(T)$ holds.

The main reason for this enlargement is presented in Figure 2.8. Unfortunately, there is no one to one correspondance between the number of agents required for a contiguous strategy and the size of the frontier of connected progressive crusade. For example, in Figure 2.8 we require 10 agents for successively cleaning $C = (\emptyset, \{e_1\}, \{e_1, e_2\})$ but the frontier of C is 7. If we enlarge the tree by artificial edges, we get the required correspondance for $C' = (\emptyset, \{e_1'\}, \{e_1', e_1''\}, \{e_1', e_1'', e_2'\}\{e_1', e_1'', e_2', e_2''\})$. Here $w(\{e_1', e_1'', e_2'\}) = 10$ gives the frontier.

Lemma 20 *Let T' be a tree so that every link has at least one vertex of degree 2. If there is a progressive connected crusade of frontier $\leq k$ in T', there is a monotone contiguous search strategy using $\leq k$ guards in T' and the guards can be initially placed at a single vertex v_1.*

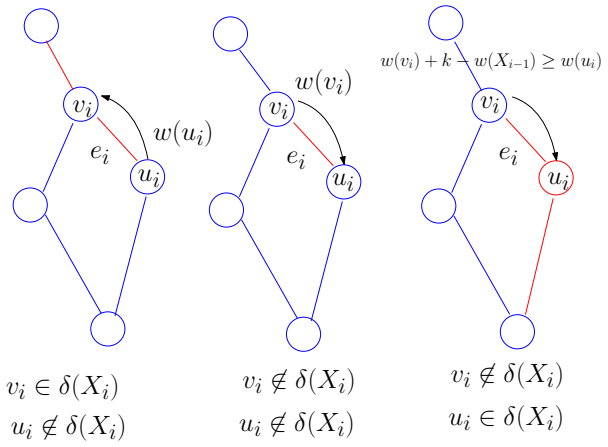

Figure 2.10: Not both vertices v_i and u_i can be in $\delta(X_i)$. Therefore, also edge e_i can be cleaned.

Proof. For the progressive connected crusade $C = (X_0, X_1, \ldots, X_m)$ of frontier $\leq k$ and $e_i = (v_i, u_i) := X_i \setminus X_{i-1}$ we construct a strategy that clears the links $e_1, \ldots, e_m$ successively.

In the beginning we set k guards at v_1. We have $w(X_1) = w(v_1) + w(u_1) \leq k$ and also $w(e_1) \leq w(u_1)$ and move $w(u_1)$ searchers along w_1.

Now by induction let us assume that we have constructed a monotone contiguous strategy for $i-1$ links $e_1, \ldots, e_{i-1}$ without recontaminations. Consider the edge $e_i = (v_i, u_i)$ incident to X_{i-1}, say $v_i \in \delta(X_{i-1})$.

If $w(X_{i-1}) + w(u_i) \leq k$ holds, $w(u_i)$ agents can move from v_i along e_i to u_i and clears link e_i and we are done.

Therefore, assume that $w(X_{i-1}) + w(u_i) > k$ holds. In this case not both vertices of e_i can be in $\delta(X_i)$. We have $v_i \in \delta(X_{i-1})$. Assume $v_i \in \delta(X_i)$. We conclude $\deg(v_i) > 2$ and $\deg(u_i) = 2$. This means that $u_i \in \delta(X_i)$ implies that the other link $f_i \neq e_i$ that contains u_i has to be contaminated, and $u_i \notin \delta(X_{i-1})$. Therefore, $w(X_i) = w(X_{i-1}) + w(u_i)$, but this has to be $\leq k$, a contradiction.

For $w(X_{i-1}) + w(u_i) > k$ at most one vertex of (v_i, u_i) is in $\delta(X_i)$ and we consider the corresponding cases; see also Figure 2.10.

1. $v_i \in \delta(X_i), u_i \notin \delta(X_i)$**:** As shown above $\deg(u_i) = 2$ and the other link $f_i \neq e_i$ that contains u_i belongs already to X_{i-1}. There are already $w(u_i)$ guards at u_i in step $i-1$ and they can clear the edge from u_i to v_i.

2. $v_i \notin \delta(X_i), u_i \notin \delta(X_i)$**:** This means that e_i is the only contaminated edge adjacent to v_i and u_i. We can move with $w(v_i)$ searchers from v_i to u_i.

3. $v_i \notin \delta(X_i), u_i \in \delta(X_i)$**:** This means that $w(X_i) = w(X_{i-1}) - w(v_i) + w(u_i)$ and we have at least $w(v_i)$ guards at v_i. Move all $k - w(X_{i-1})$ free guards to v_i. We have $w(v_i) + k - w(X_{i-1}) \geq w(v_i) + w(X_i) - w(X_{i-1}) \geq w(u_i)$ agents at v_i that can clear e_i now.

Altogether, there is a successful montone contiguous strategy for T' with k agents adapted from the progressive connected crusade $C = (X_0, X_1, \ldots, X_m)$ of frontier $\leq k$. □

Note, that the above Lemma also holds for graphs with the same properties. The same holds for the following Lemma where we show that we can obtain a strategy for T from the strategy for T'.

Lemma 21 *Any contiguous monotone strategy for T' can be translated to a contiguous monotone strategy for T with the same number k of agents.*

Proof. Let $e' = (x, y)$ and $e'' = (y, z)$ be links stemming from the extension of a link e. If q guards move from x to y or z to y, they stay in there place in T. If q guards move from y to x or from y to z, they will move from z to x or from x to z in T, respectively. □

The other way round, any strategy for T is also a strategy for T'.

Lemma 22 *Any contiguous monotone strategy for T with k agents can be translated to a contiguous monotone strategy for T' with the same number k of agents.*

Proof. A move along an edge e in T is splitted into two moves along e' and e'' in T'. If the move clears e, then $q \geq w(e)$ have traversed e. From the construction q searchers are also enough for $w(e) = w(e') = w(e'')$ and the weight $w(e)$ of the intermediate vertex. □

We collect our results:

Proof of Theorem 17: From Lemma 21 we conclude $\mathrm{cs}(T') \leq \mathrm{cs}(T)$. From Lemma 18 we obtain a connected crusade of frontier $\leq \mathrm{cs}(T)$ in T'. From Lemma 19 we conclude that there is a progressive connected crusade of frontier $\leq \mathrm{cs}(T)$ in T'. From Lemma 20 we obtain a monotone contiguous search strategy using $\leq \mathrm{cs}(T)$ guards in T' and we can assume that all searchers are initially at a single starting vertex v_1. From Lemma 22 we conclude that there is also an optimal monotone contiguous search strategy that starts with all guards in a single vertex.

2.2.5 Designing a monotone strategy for unit weights

By Theorem 17 we can start strategy from a single vertex v and we can consider monotone strategies. Therefore, we design an optimal strategy for any starting vertex v and for the rooted tree T_v we compute the minimum number, $\mathrm{cs}(T_v)$, of agents required for starting in v. Finally we have $\mathrm{cs}(T) = \min_{v \in T} \mathrm{cs}(T_v)$.

An optimal monotone strategy for computing, $\mathrm{cs}(T_v)$, will also give an ordering all vertices z of T_v, stating which subtree, say $T_v(z)$, of T_v w.r.t. root v is fully cleared first. For this we can also consider the subtree $T_v(z)$ alone with root z and ask for $\mathrm{cs}(T_v(z))$ for short and an optimal monotone strategy.

We denote the children of the vertex z of the subtree $T_v(z)$ of T_v by $z_1, \ldots, z_d$ w.r.t. the order $\mathrm{cs}(T_v(z_i)) \geq \mathrm{cs}(T_v(z_{i+1}))$ for $i = 1, \ldots, d-1$. An example is given in Figure 2.11. Now, we can prove the main structural result. Unfortunately, there is a flaw in the proof of Barrière at al. and we can only proof the statement for unit weighted trees. The flaw is precisely marked in the proof below.

Lemma 23 *Let $z_1, \ldots, z_d$ be the $d \geq 2$ children of a vertex z in T_v and assume that $cs(T_v(z_i)) \geq cs(T_v(z_{i+1}))$ for $i = 1, \ldots, d-1$. We have*

$$cs(T_v(z)) = \max\{cs(T_v(z_1)), cs(T_v(z_2)) + w(z)\} \tag{2.5}$$

it the tree T is a tree with unit weights.

Proof. We can assume that $\mathrm{cs}(T_v(z)) \geq \mathrm{cs}(T_v(z_1))$ holds because we have to clear $T_v(z_1)$ before clearing $T_v(z)$. If in Equation 2.5 $\mathrm{cs}(T_v(z_1) \geq \mathrm{cs}(T_v(z_2) + w(z)$ holds, we can clear $T_v(z)$ by setting $w(z)$ on z and clear all $T_v(z_i)$ by $\mathrm{cs}(T_v(z_1)$ agents but $T_v(z_1)$ last. Note that also $w((z, z_i)) \leq w(z_i) \leq \mathrm{cs}(T_v(z_i)$ for all i for moving back from subtrees to z. Altogether, $\mathrm{cs}(T_v(z_1)$ agents are required and they are sufficient.