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!

2. If not all subtrees from 2^{α_1-2} to 2^0 exist, only α_1-1 agents are required. But now the value for n is small enough so that we can conclude. $\alpha_1 - 1 \geq \lfloor \log_2(\frac{2}{3}(n+1)) \rfloor \geq \log_2(n+1) - 1$ which gives the bound. This requires the measurement of $\frac{2}{3}(n+1)$ in comparison to $\alpha_1 - 1$ is left as an Exercise.

□

Exercise 14 *Discuss the remaining case in the above proof. That is $\alpha_2 - 1 < \alpha_1$ and the two cases depicted in the proof.*

On the other hand, we show that $\lfloor \log_2 n \rfloor$ agents are always sufficient.

Lemma 30 *For every $n \geq 1$ and unit weights, $\lfloor \log_2 n \rfloor$ agents are sufficient for a contiguous search strategy.*

Proof. We consider a tree T_r with n vertices and $\mu(r) = \text{cs}(T)$. Now, we simplify this so that it becomes a complete binary tree T'_r w.r.t. r with $\text{cs}(T_r) = \text{cs}(T'_r)$ by the following rules. The rules can be applied successively, until none of them is applicable any more. The children/parent relation in the tree is considered w.r.t. r.

1. For a node x and its $d > 2$ children $x_1, x_2, \ldots, x_d$ ordered by $\text{cs}(T_r(x_i)) \geq \text{cs}(T_r(x_{i+1}))$ remove all $T_r(x_i)$ for $i > 2$.
2. For a node x with two children x_1 and x_2 and $\text{cs}(T_r(x_1)) > \text{cs}(T_r(x_2))$, remove $T_r(x_2)$.
3. For a node $x \neq r$ with only one child x_1, remove x and connect x_1 to the parent of x.
4. If there are more than two vertices left, and r has only one child x_1, remove x_1 and connect the children of x_1 to r.

First, the number of agents required for T'_r and T_r are the same, because the computation of $\mu(r)$ in T_r makes use of exactly the same values. Note that the weights of the vertices are restricted to one, therefore rule 2. is also correct by $\text{cs}(T_r(x_1)) \geq \text{cs}(T_r(x_2)) + 1$. Cancelling a vertex with one child has no influence.

Second, we show that T'_r is a complete binary tree rooted in r. The first rule and the second rule returns a tree that has internal nodes with at most 2 children. Rule three deletes internal nodes with one child except for the root. Rule 4 makes the root have 2 or 0 children.

Thus, we have a binary tree whose internal nodes have degree excactly 2. Finally, we show that the tree is complete. Let x be a node such that the subtree T'_x at x is not complete and there is no other subtree in T'_x with this property. This means that the children x_1 and x_2 of x in T'_r define complete subtree T'_{x_1} and T'_{x_2} of different size. Thus, rule 2 can be applied which gives a contradiction. □

2.2.8 The prize of connectivity

In the previous section we analyzed the contiguous search number for trees and presented a polynomial time algorithm for trees. The key argument was that recontamination does not help for decreasing the search number. The contiguous search idea is mainly based on the fact that searchers should not jump.

In general in the non-continuous setting this is in some sense allowed. More precisely, we extend the rules defined in the beginning of Section 2.2.3. We allow that some of the agents can be retracted from somewhere and placed somewhere else.

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

We consider the unit-weighted case in this section. Note that the monotonicity proof in the previous section also holds for non-contiguous strategies for trees and also for graphs. And it also holds, if Rule 3. can be applied. This means that the progressive crusades of frontier k and the search number k for graphs correspond in general in the same way. Recontamintion does not help and optimal monotone strategies always exists.

More precisely, the connectivity relationship in the proof of Lemma 19 depicted in Figure 2.7 was only shown for trees. We obtained a progressive *connected* crusade. In general the use of a progressive crusade is sufficient. Conversely, in the proof of Lemma 20 the three cases depicted in Figure 2.10 can also be handled, if the progressive crusade is not connected.

Exercise 15 *Consider the proof of Theorem 17. Argue, that with the same arguments, we can show:* For any unit-weighted graph G, with search number $s(G)$, there is always a monotone strategy with $s(G)$ searchers.

So we can ask what is the prize for the connectivity. General strategies for the above rules indeed have better search numbers as we will show here. We between the search number, $s(G)$, for general strategies and the contiguous search number, $\mathrm{cs}(G)$, for contiguous strategies. As mentioned above for both measures we find optimal monotone strategies.

Let D_k denote a tree with root r of degree three and three full binary trees, B_{k-1}, of depth $k-1$ connected to the r. We first show that $cs(D_k) = k + 1$ holds.

Lemma 31 *For the graph D_k, we conclude $cs(D_k) = k + 1$.*

Proof. Let T_1, T_2 and T_3 denote the copies of B_{k-1} connected to the root and let e_i denote the edge that connects T_i with the root r. For the contiguous search w.l.o.g. we can assume that the edge e_1 is cleared first at timestep i_1 among the edges e_1, e_2 and e_3. W.l.o.g. let $i_2 > i_1$ denote the time step where for the first time a leaf l of T_2 or T_3 is reached. Assume w.l.o.g. $f \in T_2$. At time step $i_2 - 1$ the path $P(r, x_k) = r, x_1, \ldots, x_{k-1}$ of length $k - 1$ from r to the neighbor x_{k-1} of f with k vertices has to be clean and for any $x_i \in P(r, f)$ there is a unique subtree in T_2 different from f that is not fully decontaminated. For the root r there is a subtree in T_3 that is not fully decontaminated. So at least one searcher for any x_i and for r is required which gives k in total. One additional searcher now is required for cleaning f. This gives $\mathrm{cs}(D_k) \geq k + 1$.

On the other hand $k + 1$ searchers are sufficient, if we start at the root with $k + 1$ agents and first clean a leaf and its neighbor. Recursively, and full binary subtree of depth l is cleaned from the root with $l + 1$ searchers. □

Now, we would like to relate this to the number $s(D_k)$. We consider D_{2k-1} with $\mathrm{cs}(D_{2k-1}) = 2k$.

Lemma 32 *For D_{2k-1} we conclude $s(D_{2k-1}) \leq k + 1$.*

Proof. For $k = 1$ the statement is trivial. So assume $k > 1$. We first place one agent at the root r and successively clean the copies of B_{2k-2} by k agents. The last statement is shown by induction. For $k = 2$ we place one agent at the root and a single agent starting at a leaf cleans first the left subtree of r, is then moved to the leaf of the right subtree and cleans the second subtree. Finally, all agents are placed at the root.

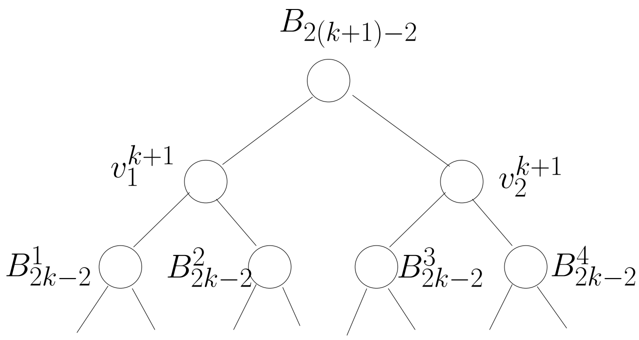

Figure 2.17: The inductive step. Each subtree B^i_{2k-2} can be cleaned by k agents. Placing an additional searcher at v_1^{k+1} in the beginning, we first clean B^1_{2k-2} and B^2_{2k-2} successively by k agents, move all $k+1$ searchers to v_2^{k+1} and do the same for B^3_{2k-2} and B^4_{2k-2} successively. Altogether, $k+1$ searchers are sufficient for $B_{2(k+1)-2}$.

Let us assume that the statement holds for $k \geq 2$. We can fully clean B_{2k-2} with k searchers that are finally located at the root. The tree $B_{2(k+1)-2}$ has four subtrees B^i_{2k-2} for $i = 1, 2, 3, 4$ of depth $2k - 2$as shown in Figure 2.17 that can be cleaned by induction hypothesis with k agents. Let v_1^{k+1} be the ancestor of B^1_{2k-2} and B^2_{2k-2} and let v_2^{k+1} be the ancestor of B^3_{2k-2} and B^4_{2k-2}.

We place one additional agent at v_1^{k+1} and clean B^1_{2k-2} and B^2_{2k-2} successively by k agents. Then all $k+1$ agents move over the root toward v_2^{k+1}. Here we again leave the additional agent at v_2^{k+1} and clean clean B^3_{2k-2} and B^4_{2k-2} successively by k agents. Thus $k+1$ agents are required.

By induction, B_{2k-2} can be cleaned by k searchers and for D_{2k-1} at most $k+1$ searchers are required. □

Now, we have a fixed relationship between $cs(G)$ and $c(G)$ for $G = D_{2k-1}$. We have $s(D_{2k-1}) \leq k+1$ and $\mathrm{cs}(D_{2k-1}) = 2k$.

Corollary 33 *There exists a tree T so that $cs(T) \leq 2c(T) - 2$ holds.*

It was also shown by Barrière et al. 2012, that there is no tree T with ratio $\frac{cs(T)}{c(T)}$ larger than 2. More precisely,

$$\frac{cs(T)}{c(T)} < 2 \text{ for all trees } T\,.$$

The proof of this fact relies on the fact that in principle (up to retractions) the trees D_k can be considered to be the category of graphs that gives the worst-case ratio. If there is some time left at the end of the semester we will prove this fact.