Server Placements, Roman Domination and Other Dominating Set Variants[*]

Aris Pagourtzis[†] Paolo Penna[‡] Konrad Schlude[‡]
Kathleen Steinhöfel[§] David Scot Taylor[‡] Peter Widmayer[‡]

Abstract

Dominating sets in their many variations model a wealth of optimization problems like facility location or distributed file sharing. For instance, when a request can occur at any node in a graph and requires a server at that node, a minimum dominating set represents a minimum set of servers that serve an arbitrary single request by moving a server along at most one edge. This paper studies domination problems for two requests. For the problem of placing a minimum number of servers such that two requests at different nodes can be served with two different servers (called win-win), we present a logarithmic approximation, and we prove that nothing better is possible. We show that the same is true for Roman domination, the well studied problem variant that asks for each vertex to either possess its own server or to have a neighbor with two servers. Still the same is true if each idle server can move along one edge while the first of both requests is being served. For planar graphs, we propose a PTAS for Roman domination (and show that nothing better exists), and we get a constant approximation for win-win.

1 Introduction

In this paper, we study a generalization of the *dominating set problem* [GJ79]. We are given a graph, and at every node of this graph a request

[*] This work was partially supported by the Swiss Federal Office for Education and Science under the Human Potential Programme of the European Union under contract no. HPRN-CT-1999-00104 (AMORE).

[†] Department of Computer Science, University of Liverpool, United Kingdom. E-mail: aris@csc.liv.ac.uk. (Partially supported by EPSRC grant GR/N09855.)

[‡] Institute of Theoretical Computer Science, ETH Zentrum, CH-8092 Zürich, Switzerland, E-mail: {lastname}@inf.ethz.ch.

[§] GMD - FIRST, Berlin, Germany. E-mail: Kathleen.Steinhoefel@gmd.de

can appear. We want to service such requests. To do so, we place servers at nodes. The request at a node v is serviced, if there is a server on v, or if a server in its neighborhood is moved to v. Clearly, if we want to be able to service one request, then the multiset of server locations must contain a dominating set of nodes.

However, there are applications in which we want to ensure that more than one request can be serviced. In this paper, we study the case of two requests. Imagine, e.g., that two requests occur simultaneously and a server can satisfy only one at a time. We view our problem as a member of the large family of dominating set problems, of which [HHS98] already cite more than 75 different variants. These may depend on conditions on the dominating set DS (e.g. connectivity) or on the other nodes (e.g. a node is dominated if there is a node in DS at distance at most k, or each vertex is dominated at least k times, or exactly once, etc.). The study of such dominating set problems is motivated by their applications to facility location (minimizing the number of facilities, subject to every demand being close enough to some facility), file sharing in distributed systems [NR95], game theory [dJ62], etc.

Interestingly, some very old questions have also triggered new research on the topic [AF95, RR00, Ste99]:

> Roman Domination : Where should the armies of the Roman Empire be placed so that a smallest number of armies can protect the whole Empire (see Figure 1)?

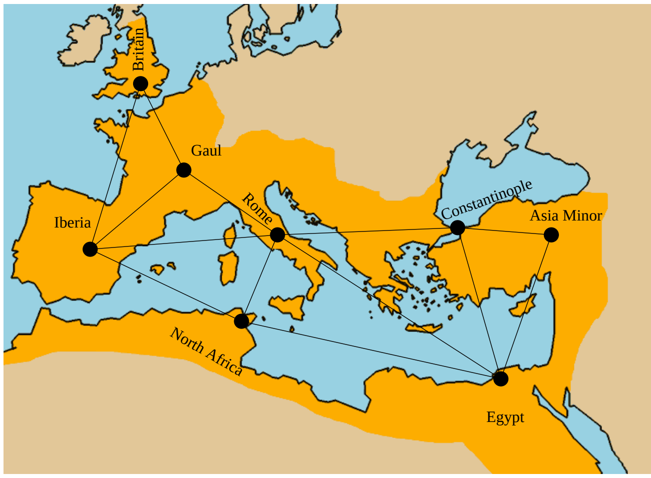

Figure 1: The Roman Empire around 300 A.C.

The assumption is that an area can be protected either by one army located inside the area, or by an army in a neighbor area that comes over for the defense. In the latter case it is required that a second army remains in the neighbor area, so that it can quickly confront a second attack. A reason for the historical 1-2 requirement (one army here or two at a neighbor) is that we want to be able to service two requests in one time unit (provided that no two requests can come from the same point at the same time).

In this paper we deal with variants of the ROMAN DOMINATING SET [Dre00, Ste99]. In particular, we consider the case in which there are two requests we want to service and no two requests appear at the same node. Moreover, a server that is used to service the first request cannot be used to service another request. A solution to our problem for a given graph is a set of servers at nodes; since all servers are identical, a multiset of nodes (where the multiplicity of a node is the number of servers at that node) represents a server placement.

Two factors we will consider are: (i) whether the two requests are known before the first one must be serviced (OFFLINE), or the first one must be serviced before the second one is known (ONLINE), and (ii) whether servers must stay in place unless they service a request (STATIC), or we allow for a rearrangement (DYNAMIC): as one server services the first request, all other servers are allowed to move to a neighbor node. The goal of the move is to guarantee that any second request can be handled, too, in the ONLINE case (that is, the resulting server placement is a dominating set if we ignore the first requesting node and its server). The ONLINE STATIC WIN-WIN version has been discussed earlier [Och96] and called *Win-Win* there. (Unlike in Roman Domination, in this case we only require to be able to win against any two consecutive attacks.) Since our problems also deal with two consecutive requests, we adopt the name terminology and we denote the four problem variants as ONLINE STATIC WIN-WIN, ONLINE DYNAMIC WIN-WIN, OFFLINE STATIC WIN-WIN, and OFFLINE DYNAMIC WIN-WIN.

1.1 Our (and Previous) Results

In this paper we investigate the relationships between the above problems (including ROMAN DOMINATION), as well as the complexity of computing exact and approximate solutions. In particular, we consider the following questions:

1. Given a multiset S, is S a feasible solution to (one of) the above

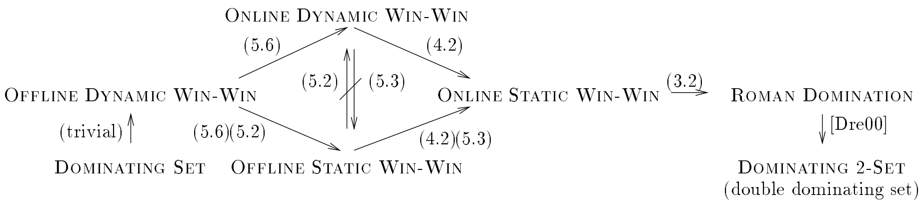

Figure 2: Relationships between the problems: arrows represent '$\prec$' and they are numbered according to the corresponding theorem.

problem variants? Is there a combinatorial characterization for those S?

2. Let varA Win-Win and varB Win-Win denote any two problem variants. If S is a solution for varB Win-Win, does this imply that S is also a solution to varA Win-Win?

3. A positive answer to the above question implies that $\mathsf{opt}_{\mathrm{A}}(G) \leq \mathsf{opt}_{\mathrm{B}}(G)$, where $\mathsf{opt}_{\mathrm{A}}$ and $\mathsf{opt}_{\mathrm{B}}$ denote the minimum size multiset solving the two variants, resp. Is there a graph for which the inequality is strict?

Let varA Win-Win$\preceq$ varB Win-Win denote the fact that Question 2 has a positive answer, and let varA Win-Win$\prec$ varB Win-Win denote the fact that Question 3 does too. It turns out that the problems we look at form the partial order in Figure 2.[1] Noticeably, this relationship also holds when we restrict ourselves to planar graphs.

As for Question 1, for two out of the four win-win problems we provide a characterization of those multisets corresponding to each problem. For the Dynamic Win-Win, we prove the NP-hardness of the rearrangement step after the first request. This result seems to denote that such a characterization for this problem version does not exist, or at least is different from those given for the other two problems (those can be checked in polynomial time).

This leads us to complexity and (non-) approximability issues. Intuitively, the $\preceq$ relationship may have some consequences on the (non-) approximability of those problems. Indeed, the order in Figure 2, combined with the fact that "doubling" a dominating set (the Dominating 2-Set problem in Figure 2) yields a feasible solution for *all* of the problems, implies an approximation preserving reduction ($\leq_{\mathsf{AP}}$, see [ACG^{+}99]) between

[1] Figure 2 contains a new problem (Dominating 2-Set) which we introduce to prove some of our results.

Problem Version	**General Graphs**	**Planar Graphs**
Roman Domination	$(2+2\ln n)$-APX, not $c \log n$-APX (NP-hard [Dre00])	PTAS, in P for r-outerplanar (NP-hard [Hed00]) (in P for trees & $(r \times n)$-grids [Dre00])
Online Sta. Win-Win, Online Dyn. Win-Win, Offline Sta. Win-Win	$(2+2\ln n)$-APX, not $c \log n$-APX	$(2+\epsilon)$-APX, for any $\epsilon > 0$

Table 1: Hardness and approximability: Our and previous results. (All NP-hardness results are in strong sense, thus implying the non-existence of a FPTAS. Previous results are displayed between brackets.)

all these problems. Let $f(n)$-APX denote the class of problems that admit a polynomial-time $f(n)$-approximation algorithm [ACG$^+$99]. In Table 1 we summarize the complexity and (non-) approximability results of this work.

As for the results on planar graphs, our technical contribution is a Polynomial-Time Approximation Scheme (PTAS) for Roman Domination. This result is based on an exact polynomial-time algorithm for r-outerplanar graphs. The latter improves over the previous results in [Dre00]: in this work only trees and $r \times n$-grids (for any *fixed* r) are shown to be exactly solvable. Our result subsumes both of them (an $r \times n$-grid is clearly an r-outerplanar graph).

2 Online Static Win–Win

In the sequel, given a multiset S, $\mathrm{uniq}(S)$ denotes the set resulting by removing multiplicities.

Definition 2.1 (online static) *Given a graph $G = (V, E)$, a* server placement *for G is a multiset S of nodes. A server placement S is a* win–win *for G, if for all $v \in V$ there is an $u_v \in S$ with the properties:*

1. *$v = u_v$ or $(u_v, v) \in E$,*
2. *for all $v' \in V \setminus \{v\}$ there is an $u_{v'} \in S \setminus \{u_v\}$ with*

$$v' = u_{v'} \text{ or } (u_{v'}, v') \in E.$$

Lemma 2.2 (sandwich) *Any graph G has the following properties:*

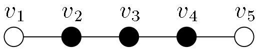

Figure 3: A win–win.

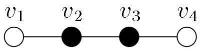

Figure 4: Not a win–win.

1. *For every dominating set DS, the server placement $SP := DS \uplus DS$ is a* win–win *for G, where $\uplus$ denotes the multi-union.*
2. *For every* win–win *WW, the set* $\text{uniq}(WW)$ *is a dominating set of G.*
3. *For every minimum dominating set MDS and for every minimum* win–win *MWW, $|MDS| \leq |MWW| \leq 2|MDS|$ hold.*

Proof. For Property 1, let $v_1, v_2 \in V$ be a pair of nodes with $v_1 \neq v_2$. Since DS is a dominating set, there are $u_{v_1}, u_{v_2} \in DS$, such that

$$\begin{aligned} v_1 &= u_{v_1} \text{ or } (v_1, u_{v_1}) \in E, \text{ and} \\ v_2 &= u_{v_2} \text{ or } (v_2, u_{v_2}) \in E \end{aligned}$$

hold. Due to the definition of SP, $\{u_{v_1}, u_{v_2}\} \subset SP$ holds. This implies that requests at v_1, v_2 can be serviced.

For Property 2, let $v \in V$ be a node. There is a $u_v \in WW$ with $v = u_v$ or $(v, u_v) \in E$. Since $u_v \in \text{uniq}(WW)$, $\text{uniq}(WW)$ is a dominating set.

For Property 3, it suffices to consider the win–win $WW := MDS \uplus MDS$ and the dominating set $DS := \text{uniq}(MWW)$. Clearly, $|MDS| \leq |DS| \leq |MWW| \leq |WW| = 2|MDS|$. □

2.1 Characterization of win–win Multisets

The property of being a win–win does not depend only on a node and its neighbors. Furthermore, it is not enough that for every pair of nodes there are two different adjacent servers. This is illustrated by the example in Figure 4. The server placement $S = \{v_2, v_3\}$ is not a win–win. If the first request is at v_2, then there are two cases. Case 1, the request is serviced by v_2, then a second request at v_1 cannot be serviced. Case 2, the request is serviced by v_3, then a second request at v_4 cannot be serviced.

This observation lead us to the following characterization of the server placements that are win–win.

Definition 2.3 *Given a graph $G(V, E)$ and a multiset D for it, a vertex $v \in V$ is* weak *if D dominates v only once. A vertex $u \in D$ is* safe *if every $v \in N(u)^+$ is not weak, where $N(u)^+ = N(u) \cup \{u\}$.*

Lemma 2.4 *A multiset D for $G(V, E)$ is a* win–win *if and only if the following two properties hold:*

at-most-1-weak *Every $u \in D$ does not dominate more than one weak node;*

at-least-1-safe *Every non weak node $v \in V$ is dominated by at least one safe node $u \in D$.*

Proof.
($\Rightarrow$) By contradiction, assume that some $u \in D$ does not satisfy Property at-most-1-weak. Then, there exist two weak nodes w_1 and w_2 dominated *only* by u. After a first request at w_1, w_2 is no longer dominated (we must have used u for the first request). This contradicts the hypothesis that D is a win–win. Now suppose (again by contradiction) that a non weak node v is not adjacent to any safe node (thus contradicting Property at-least-1-safe). Let $u_1, \ldots, u_k$ be the nodes of D adjacent to v, for some $k \geq 2$ (this follows from the fact that v is not weak). By hypothesis, none of $u_1, \ldots, u_k$ is safe. So, there exist $w_1, \ldots, w_k$ distinct weak nodes, with w_i adjacent to u_i, for $1 \leq i \leq k$. Now consider a first request at node v. For this request we must use one among $u_1, \ldots, u_k$, let us say u_j. Then, if the second request is at the weak node w_j we do not have any server to react. Again, this contradicts the hypothesis.
($\Leftarrow$) Let v_1 be the position of the first request. We have two cases: v_1 is weak, or v_1 is not weak. In the first case, we must use the only node $u \in D$ that is adjacent to v_1; Property at-most-1-weak guarantees that every node in $N(u)^+ \setminus \{v_1\}$ will still be dominated. So, any second request can be handled. Otherwise, that is, v_1 is not weak, Property at-least-1-safe implies that there exists a $u \in D$ which is safe; we use such a u for this request. At this point all the nodes in $N(u)^+ \setminus \{v_1\}$ will still be dominated by some $u' \in D$. Also in this case any second request can be handled. □

2.2 Complexity

We are interested in the complexity of the Online Static Win-Win problem. We discuss hardness and approximation of this problem. Both NP-hardness and approximation hardness can be proved using the following lemma.

Lemma 2.5 *Any $f(n)$-approximation algorithm A for* Min Dominating Set *implies a $2f(n)$-approximation algorithm for* Min Online Static Win-Win. *Conversely, any $g(n)$-approximation algorithm B for* Min Online

Static Win-Win *implies a* $2g(n)$*-approximation algorithm for* Min Dominating Set*.*

Proof. Applying A to any graph G we can find a dominating set DS of size $|DS| \leq f(n)|MDS_G|$. By Lemma 2.2 the server placement $SP = DS \uplus DS$ is a win–win for G of size $|SP| = 2|DS| \leq 2f(n)|MDS_G| \leq 2f(n)|MWW_G|$.

Conversely, applying B to any graph G we obtain a win–win SP of size $|SP| \leq g(n)|MWW_G|$. Then, according to Lemma 2.2 the set $DS = \text{uniq}(SP)$ is a dominating set of size $|DS| \leq |SP| \leq g(n)|MWW_G| \leq 2g(n)|MDS_G|$. □

We know that Min Dominating Set is not approximable within $c \log n$ for some $c > 0$ [RS97] (unless P=NP) and that it is approximable within $1 + \ln n$ [Joh74]. From these facts and the above lemma one can easily prove the following.

Theorem 2.6 *The* Min Online Static Win-Win *problem in general graphs can be approximated within* $2 + 2 \ln n$*, but (unless* P=NP*) cannot be approximated within* $c \log n$ *for some* $c > 0$*.*

For Min Dominating Set in planar graphs a Polynomial Time Approximation Scheme (PTAS) is known [Bak94]. Therefore, Lemma 2.5 implies an approximation algorithm for Min Online Static Win-Win in planar graphs, called Min Planar Online Static Win-Win, with ratio $2 + \epsilon$ for every $\epsilon > 0$.

Moreover, this approximation ratio is tight for the approach of "doubling" a dominating set to construct the solution. We illustrate this by the example in Figure 5. For this graph, the set $M := \{v_1, \ldots, v_8\}$ is a minimum dominating set. Doubling it gives a solution WW with $|WW| = 16$. On the other hand, the server placement $MWW = \{w, v_1, v_2, \ldots, v_8\}$ is a minimum win–win with $|MWW| = 9$. In this case, the approximation ratio is $16/9$. If we increase the number of rays from 8 to k, then we get $|WW|/|MSP| = 2k/(k+1)$. This shows that there exist graphs for which the simple doubling algorithm has approximation ratio greater than $2 - \epsilon$, for any $\epsilon > 0$.

3 Roman Domination

We come back to the original problem of the so called Roman Domination. On every node, we can place none, one, or two servers.

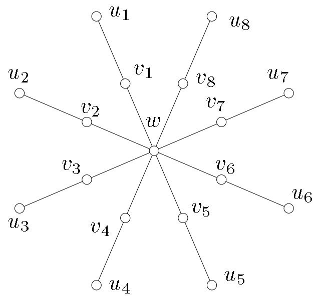

Figure 5: Doubling a dominating set gives a win–win of cost roughly twice the optimum.

Definition 3.1 (roman domination) *Given a graph $G = (V, E)$, a* roman *for G is a server placement S such that every node v in V either belongs to S or has a neighbor u in S whose multiplicity in S is at least 2. Formally, $\forall v \in V, v \notin S \to \exists u : (v, u) \in E \wedge \{u, u\} \subset S$.*

Clearly, every roman S is a win–win: If the first request is at a node $v \in S$, then v is serviced by its own server; if $v \notin S$, then v is serviced by a neighbor u with $\{u, u\} \in S$. This implies that a minimum win–win does not have cardinality larger than a minimum roman. The next result shows that the '$\preceq$' relationship between those two problems is actually strict:[2]

***Strict Inclusion* 3.2** Online Static Win-Win$\prec$ Roman Domination*:* For the graph in Figure 3, the server placement $S' = \{v_2, v_2, v_4, v_4\}$ is a minimum roman. On the other hand, $S = \{v_2, v_3, v_4\}$ is a minimum win–win: if the first request is at v_2, then this request is serviced by v_3; if after that the second request is at v_3, then it is serviced by v_2 or by v_4. $\diamondsuit$

It is known that Min Roman Domination is NP-hard for arbitrary graphs [Dre00]. We strengthen this result and show that the problem is also hard to approximate. As a by–product, we get a new proof for the NP-hardness. In particular, Lemma 2.2 remains true if we replace the notion of win–win by roman (see also [Dre00, Proposition 2.1]). Hence, we get the following theorem:

Theorem 3.3 *The* Min Roman Domination *problem in general graphs can be approximated within $2 + 2 \ln n$, but (unless $\mathsf{P} = \mathsf{NP}$) cannot be approximated within $c \log n$ for some $c > 0$.*

[2] Since in all cases '$\preceq$' is trivial, in the sequel we will only show that '=' does not hold.

3.1 Planar Graphs

Often, our problem instances are not arbitrary graphs; planarity is quite a natural condition (see Figure 1). It is therefore interesting to study the problem complexity for planar graphs, since we know that minimum dominating set can be approximated well for planar graphs. It turns out that Min Roman Domination is NP-hard for planar graphs.[3]

Theorem 3.4 Min Roman Domination *is strongly* NP*-hard even if the input graph G is planar.*

Proof Sketch. We show the NP-hardness of Min Roman Domination by reducing Planar Vertex Cover [GJ79]. Indeed, in an adaptation of the well known reduction from vertex cover to dominating set, we can make the local transformation upon an edge in Figure 6. The resulting graph, with $|V| + 2|E|$ vertices and $5|E|$ edges is still planar, and it is straightforward to show that a vertex cover with k nodes in the original graph exists if and only if a roman with $2k$ nodes exists in the second. □

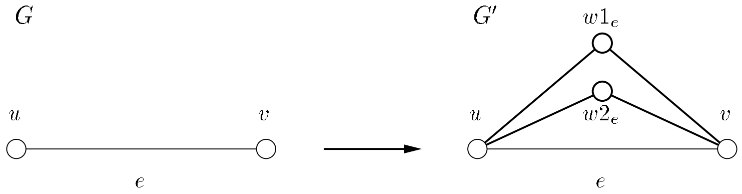

Figure 6: Reduction from Planar Vertex Cover to Roman Domination.

The results from the previous section show that the planar Min Roman Domination can be approximated within $2 + \epsilon$. The next theorem shows that we can find a better approximation. Its proof follows the ideas from [Bak94, ABFN00] which have become a well known standard method to get PTASs for many problems on planar graphs. Those approximations schemes look very similar; the only specific part is that the problem has to be solved optimally on r–outerplanar graphs. We use dynamic programming and the

[3] In [Dre00, page 68], the NP-hardness of the planar graph case is also mentioned. At the writing time the paper cited in [Dre00] is unpublished, so for the sake of completeness, we include a reduction from vertex cover. This reduction is also used to prove the "tightness" of our approximability results.

notion of *bounded treewidth* [ABFN00] to show how this can be done for the MIN ROMAN DOMINATION problem.

Theorem 3.5 (PTAS) MIN PLANAR ROMAN DOMINATION *has a Polynomial Time Approximation Scheme* (PTAS)*, but (unless* $\mathsf{P} = \mathsf{NP}$*) it does not have a Fully Polynomial Time Approximation Scheme* (FPTAS)*.*

Proof. Let G be a r–outerplanar graph. This implies that G has a treewidth l of at most $3r-1$ ([ABFN00], Theorem 9). A tree decomposition $\langle\{X_i|i \in I\}, T\rangle$, with width at most $3r-1$ and with $|I| = O(|V|)$ of G, can be found in $O(r|V|)$ time ([ABFN00], Theorem 12).

Let $\langle\{X_i|i \in I\}, T\rangle$ be a tree decomposition for the graph $G = (V, E)$. Let $X_i = \{x_1^{(i)}, \ldots, x_{n_i}^{(i)}\}$ be a bag [ABFN00] with $n_i := |X_i|$. A number $j \in \{0, \ldots, 3^{n_i} - 1\}$ can be identified with a server placement $S_j^{(i)}$ in the following way. We write j in ternary arithmetic, i.e., $j = \sum_{\nu=1}^{n_i} 3^{\nu-1} j_\nu$, where $j_\nu \in \{0, 1, 2\}$. Every node $x_\nu \in X_i$ occurs with multiplicity j_ν in $S_j^{(i)}$.

The algorithm we will describe visits the vertices of T from the leaves to the root. For every server placement $S_j^{(i)}$ of a bag X_i, the algorithm computes a server placement $\overline{S}_j^{(i)}$ for the bags in the subtree rooted at i as a partial solution.

The dynamic programming algorithm proceeds in three steps.

Step 1: For every leaf X_i, for every $j \in \{0, \ldots, 3^{n_i} - 1\}$, we define $\overline{S}_j^{(i)} := S_j^{(i)}$.

Step 2: After this initialization, we visit the vertices of our tree decomposition from the leaves to the root. Suppose node i has a child k in the tree T. In the case that i has several children $k_1, \ldots, k_s$ in the tree T, this step has to be repeated for each child.

1. Determine the intersection $Y := X_i \cap X_k$.
2. For every server placement $S_j^{(i)}$ of X_i, we choose a server placement $S_{j'}^{(k)}$ of X_k such that the following properties hold:
   (a) ${S_j^{(i)}}_{|Y} = {S_{j'}^{(k)}}_{|Y}$.
   (b) For every $v \in X_k \setminus Y$ with $v \notin S_{j'}^{(k)}$, there is a u_v with $\{u_v, u_v\} \subset \overline{S}_{j'}^{(k)}$ and $(v, u_v) \in E$.
   (c) The number $|(S_j^{(i)} \uplus \overline{S}_{j'}^{(k)}) \setminus ({S_j^{(i)}}_{|Y})|$ is minimized.

Then, we define $\overline{S}_j^{(i)} := (S_j^{(i)} \uplus \overline{S}_{j'}^{(k)}) \setminus S_j^{(i)}{}_{|Y}$. For different $j_1, j_2 \in \{0, \ldots, 3^{n_i}\}$ with $S_{j_1}^{(i)}{}_{|Y} = S_{j_2}^{(i)}{}_{|Y}$, the same $j'_1 = j'_2$ can be chosen.

Note that, from Property 3 of a tree decomposition, we know that none of the nodes $v \in X_k \setminus Y$ will appear in a bag that has not been visited up to this point. Otherwise, such a node would also appear in X_i.

Step 3: Let X_R be the root of T, let $n := |X_R|$. Choose a $j \in \{0, \ldots, 3^n - 1\}$, such that

1. $\overline{S}_j^{(R)}$ is a roman for G, and
2. $|\overline{S}_j^{(R)}|$ is minimum.

The algorithm described above runs in time polynomial in the size of G and in 3^{3r}. Due to construction, for every vertex $i \in T$ and for every $j \in \{0, \ldots, 3^{n_i} - 1\}$, $\overline{S}_j^{(i)}$ is a smallest server placement such that property 2 (b) of step 2 is fulfilled. This implies that $\overline{S}_j^{(R)}$ is a minimum roman for G.

Finally, the strong NP-hardness proof of Theorem 3.4 implies that Roman Domination is not in FPTAS (see [GJ79] for the definition of strong NP-hardness and its implications). □

4 Online Dynamic Win–Win

In this section, we assume that after the first request, there is enough time to move the servers from one node to a neighbor before the second request occurs. This leads to the following definition.

Definition 4.1 (online dynamic) *Given a graph $G = (V, E)$ and a server placement S. A function*[4] *$r : S \to V$ is called* rearrangement *for G, S, if for every server $v \in S$*

$$r(v) = v \text{ or } (v, r(v)) \in E$$

holds. We say that S is a dynamic win–win *for G, if for every $u \in V$ there is a rearrangement r_u with the properties:*

- *There is $v \in S$ with $r_u(v) = u$, i.e., the first request at u can be serviced.*

[4] Note that different servers at a node can take different values.

- *For all $u' \in V \setminus \{u\}$, there is a $v' \in S \setminus \{v\}$ with $r_u(v') = u'$ or $(r_u(v'), u') \in E$.*

***Strict Inclusion* 4.2** ONLINE DYNAMIC WIN-WIN$\prec$ ONLINE STATIC WIN-WIN*:*
Consider the cycle of length 4, $(v_1, \ldots, v_4, v_1)$. By one hand, the server placement $S = \{v_1, v_3\}$ is a dynamic win–win. For instance, if the first request is at v_2, then this request is serviced by v_1 and v_3 moves to v_4. On the other hand, there is no server placement S' which is a win–win with $|S'| = 2$. To see this, we consider two cases. Case 1, $S' = S$. A first request at v_2 must be serviced by v_1 or v_3, let us say v_1. Then a second request occurring at v_1 cannot be serviced. Case 2, $S' = \{v_1, v_4\}$. Consider a request at v_1. If we use the server at v_1, then v_2 is no longer dominated. Similarly, using the server at v_4 leaves v_3 undominated. ◇

Again, the methods from Section 2 can be used to show the complexity of MIN ONLINE DYNAMIC WIN-WIN.

Theorem 4.3 *The* MIN ONLINE DYNAMIC WIN-WIN *problem is* NP*-hard. It can be approximated within $2 + 2\ln n$, but (unless* P = NP*) cannot be approximated within $c \log n$ for some $c > 0$.*

We know that finding a minimum dominating set is hard to do. But what happens if we are given a server placement, and are asked if the arrangement is 'close to' a dominating set – that is, if each server is allowed to move at most 1 step, can a dominating set be obtained?

Definition 4.4 *Let r be a rearrangement for $\langle G, S\rangle$; r is called* dominating rearrangement *for $\langle G, S\rangle$, if the server placement $\{r(v) | v \in S\}$ contains a dominating set for G.*

Given a graph G and a server placement S, the DOMINATING REARRANGEMENT problem asks whether there is a dominating rearrangement for $\langle G, S\rangle$.

Theorem 4.5 DOMINATING REARRANGEMENT *is* NP*-complete. This remains true, even if the input graph is planar.*

Proof. It is obvious that this problem is in NP. We use a reduction from SAT [GJ79] to show the NP-hardness.

Let F be a Boolean formula, given as a set U of variables and a collection C of clauses over U. We define a graph $G_F = G = (V, E)$ as follows (see

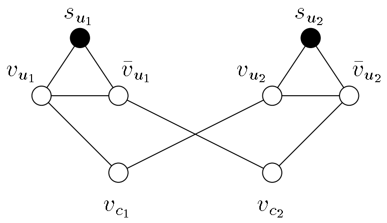

Figure 7: Reduction from SAT, $F = (u_1 \vee u_2) \wedge (\bar{u}_1 \vee \bar{u}_2)$.

Figure 7). For every variable $u \in U$, there is a *storage node* $s_u \in V$ and two *variable nodes* $v_u, \bar{v}_u \in V$. Each such triple of nodes is connected by edges, i.e., $(s_u, v_u), (s_u, \bar{v}_u), (v_u, \bar{v}_u) \in E$.

For every clause $c \in C$, there is a *clause node* $v_c \in V$. A clause node v_c is connected to a variable node v_u ($\bar{v}_u$, resp.), iff $u \in c$ ($\bar{u} \in c$, resp.). On every storage node, a server is placed, i.e., the server placement has the form $S := \{s_u\}_{u \in U}$.

For every dominating rearrangement r and for every variable $u \in U$, either $r(s_u) = v_u$ or $r(s_u) = \bar{v}_u$ hold. It is obvious, that this corresponds to a variable assignment. The given formula F is satisfiable, iff there is a dominating rearrangement for $\langle G, S \rangle$.

To prove the NP-completeness for planar graphs, we define the subgraph $G' \subset G$ by deleting the storage nodes and the adjacent edges. G' is planar, iff G is planar. It has been shown in [Lic82, Lemma 1] that SAT is NP-complete, even if the input is restricted to formulae F with the property that G' and G are planar. □

Theorem 4.6 *Given a graph G and a server placement S. The problem to decide whether S is a* dynamic win–win *for G is* NP*-complete.*

Proof. We extend the definition of the graph G in the proof of Theorem 4.5. We add a *dummy node* $v_d \in V$, and we add edges from v_d to every clause node and from v_d to every variable node. The new server placement becomes $S := \{s_u\}_{u \in U} \uplus \{v_d\}$.

If the first request is at v_d, then this request has to be serviced by v_d, since no clause node and no variable node is in S. A second request can be serviced, iff there is a dominating rearrangement. We have seen that this is a NP-complete problem. □

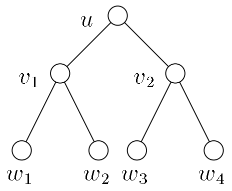

Figure 8: Proof of ***Non-Inclusion 5.3***

5 Offline Static/Dynamic Win–Win

In this section, we consider the situation in which both requests occur at the same time (equivalently, as the first request must be serviced, it is already known where the second one will be).

Definition 5.1 (offline static) *Let $G = (V, E)$ be a graph. A server placement S is an* offline win–win *if for every pair of nodes $v_1, v_2 \in V$, $v_1 \neq v_2$, there is a pair $\{u_{v_1}, u_{v_2}\} \subset S$ with*

- $v_1 = u_{v_1}$ *or* $(v_1, u_{v_1}) \in E$, *and*
- $v_2 = u_{v_2}$ *or* $(v_2, u_{v_2}) \in E$.

***Non-Inclusion* 5.2** Online Dynamic Win-Win$\not\subset$ Offline Static Win-Win*:*
For the graph in Figure 4 the set $\{v_2, v_3\}$ is an offline win–win. For the same graph, no dynamic win–win can have size 2. Indeed, consider a first request at v_2. No matter what server we use to service this request, the remaining one cannot cover the nodes $\{v_1, v_3, v_4\}$, where a second request can occur.

***Non-Inclusion* 5.3** Offline Static Win-Win$\not\subset$ Online Dynamic Win-Win*:*
It is easy to verify that $\{u, v_1, v_2\}$ is a dynamic win–win for the graph in Figure 8. On the other hand, there is no offline win–win multiset of size less than 4: each of the subtrees rooted at v_1 or v_2 must contain at least two servers. $\diamondsuit$

Again, Min Offline Static Win-Win is an NP-hard problem, illustrated by the techniques of Section 2. Moreover, we can give the following characterization of the offline win–win multisets:

Lemma 5.4 *A server placement S is an* offline win–win, *iff for every pair of two different nodes there is one server in the neighborhood of one node and a different server in the neighborhood of the other node.*

We conclude this section with OFFLINE DYNAMIC WIN-WIN. Here we combine the fact that servers can be rearranged before serving the second request (DYNAMIC) with the fact that the second request is known by the time we have to serve the first one (OFFLINE). Therefore, we have the following definition for the corresponding server placement:

Definition 5.5 (offline dynamic) *Let $G = (V, E)$ be a graph. A server placement S, is an* offline dynamic win–win *for G, if for every pair of nodes $v_1, v_2 \in V$, with $v_1 \neq v_2$, there is a pair of distinct nodes $u_{v_1}, u_{v_2} \in V$ such that v_i is at distance at most i from u_{v_i}, for $i = 1, 2$.*

***Strict Inclusion* 5.6** OFFLINE DYNAMIC WIN-WIN$\prec$ ONLINE DYNAMIC WIN-WIN*:*
Consider the cycle of length 5, $(v_1, v_2, \ldots, v_5, v_1)$. It is easy to verify that the set $S = \{v_1, v_3\}$ is an offline dynamic win–win (S is a dominating set and both servers are at distance at most 2 from any other non-server node). To prove that no multiset of size 2 can be a dynamic win–win we use the following argument. After the first request has been serviced, the set of nodes to be considered as possible positions for the second request induce a path of length 4; therefore, no matter where we place the remaining server, there is no way to dominate all such nodes. $\diamondsuit$

6 Conclusion

Clearly, these are just a few of a myriad of dominating set problems. We have looked at them individually, but have also tried to explore the connections between them. First of all, every ONLINE version is more "difficult" (i.e. requires more servers) than the corresponding OFFLINE one (i.e. $\succ$). Similarly, every STATIC problem is more "difficult" than the corresponding DYNAMIC one. Additionally, our results show that the ONLINE and the DYNAMIC features are somehow orthogonal: ONLINE DYNAMIC WIN-WIN and the OFFLINE STATIC WIN-WIN are simply not comparable.

More interestingly, we can consider more requests, or even an unbounded sequence of requests, a WIN* scenario. In this case, a server can be reused after the first time step. This problem raises interesting questions, in that the online problem looks similar to a typical online server question, but instead deals much more directly with the connectivity of dominating solutions. Another interesting difference is that instead of minimizing work, it attempts to minimize resources needed for quality of service guarantees. We will explore this relationship in a future paper.

References

[ABFN00] J. Alber, H.L. Bodlaender, H. Fernau, and R. Niedermeier. Fixed Parameter Algorithms for Planar Dominating Set and Related Problems. In *Proc. 7th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 1851 of *LNCS*, pages 97–110, 2000.

[ACG$^+$99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation – Combinatorial optimization problems and their approximability properties.* Springer Verlag, 1999.

[AF95] J. Arquilla and H. Fredricksen. Graphing an Optimal Grand Strategy. *Military Operations Research*, pages 3–17, Winter 1995.

[Bak94] B. Baker. Approximation Algorithms for NP–Complete Problems on Planar Graphs. *J. ACM*, 41(1):153–180, 1994.

[dJ62] C.F. de Jaenish. Trait des Applications de l'Analyse Mathematique au Jeau des Echecs. Petrograd, 1862.

[Dre00] P.A. Dreyer. *Applications and Variations of Domination in Graphs.* PhD thesis, Rutgers University, New Jersey, 2000.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability / A Guide to the Theory of NP–Completeness.* Freeman and Company, 1979.

[Hed00] S.T. Hedetniemi. Roman domination in graphs II. Slides and notes from presentation at 9th Quadrienn. Int. Conf. on Graph Theor., Combinatorics, Algorithms, and Applications, June 2000.

[HHS98] T.W. Haynes, S.T. Hedetniemi, and P.J. Slator. *Fundamentals of Domination in Graphs.* Marcel Dekker, New York, 1998.

[Joh74] D.S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci*, 9:256–278, 1974.

[Lic82] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[NR95] M Naor and R.M. Roth. Optimal file sharing in distributed networks. *SIAM J. Comput.*, 24(1):158–183, 1995.

[Och96] D. Ochmanek. Time to Restructure U.S. Defense Forces. *ISSUES in Science and Technology*, Winter 1996.

[RR00] C.S. ReVelle and K.E. Rosing. Defendens Imperium Romanum: A Classical Problem in Military Strategy. *American Mathematical Monthly*, pages 585–594, 2000.

[RS97] R. Raz and S. Safra. A Sub–Constant Error–Probability Low–Degree Test, and a Sub–Constant Error–Probability PCP Characterization of NP. In *Proc. 29th ACM Symp. on Theory of Computing*, pages 475–484, 1997.

[Ste99] I. Stewart. Defend the roman empire! *Scientific American*, pages 94–95, December 1999.

A Treewidth

We recall the definition of treewidth from [ABFN00].

Definition A.1 *Let $G = (V, E)$ be a graph. A* tree decomposition *of G is a pair $\langle\{X_i, | i \in I\}, T\rangle$, where each X_i is a subset of V, called a bag, and T is a rooted tree with the elements of I as nodes. The following three properties should hold:*

1. *$\bigcup_{i \in I} X_i = V$;*

2. *for every edge $\{u, v\} \in E$, there is an $i \in I$ such that $\{u, v\} \subseteq X_i$;*

3. *for all $i, j, k \in I$, if j lies on the path between i and k in T, then $X_i \cap X_k \subseteq X_j$.*

The width *of $\langle\{X_i, | i \in I\}, T\rangle$ equals $max\{|X_i| | i \in I\} - 1$. The* treewidth *of G is the minimum k such that G has a tree decomposition of width k.*

The treewidth of a graph is always bigger than 0, except for the case that $E = \emptyset$. On the other hand, the size of a bag is bounded by the number of nodes in the graph. Therefore, the treewidth of a graph is less than the number of nodes.

Example: Consider the graph G in Figure 9. We define the bags $X_1 := \{v_1, v_3\}$ and $X_2 := \{v_2, v_3\}$ and a tree T with 1 as the root and 2 as a child. The pair $\langle\{X_1, X_2\}, T\rangle$ is a tree decomposition. Since the size of both bags is 2, G has a treewidth of 1.

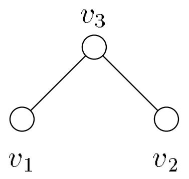

Figure 9: Graph G with treewidth 1.

G is a tree. It is easy to show that every tree has treewidth 1. The situation changes, if we add the edge $\{v_1, v_2\}$ to G, i.e., we deal with the complete graph K_3. There is no tree T', such that $\langle\{X_1, X_2, \{v_1, v_2\}\}, T'\rangle$ is a tree decomposition of K_3 (Contradiction to property 3). Therefore, the treewidth of K_3 is 2. This result can be extended, the treewidth of the complete graph K_{n+1} is n.