Roman domination:

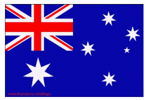
a parameterized perspective

Henning Fernau

Universität Trier 

University of Hertfordshire 

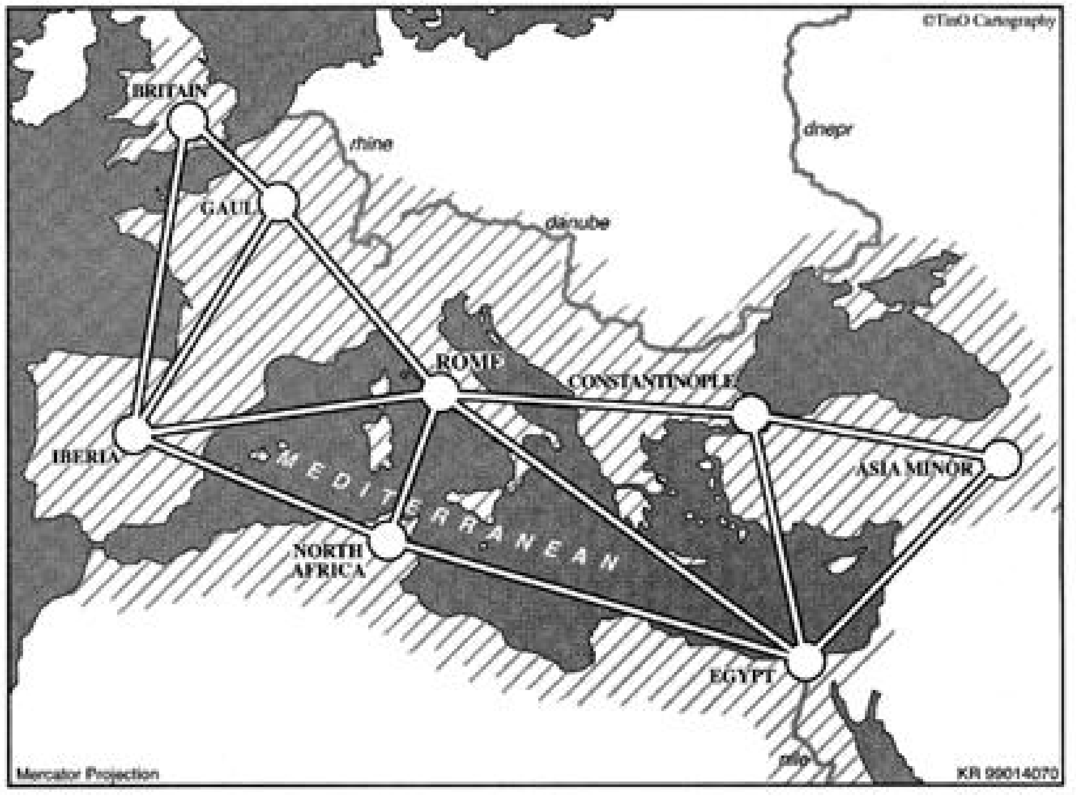
Universität Tübingen 

The University of Newcastle 

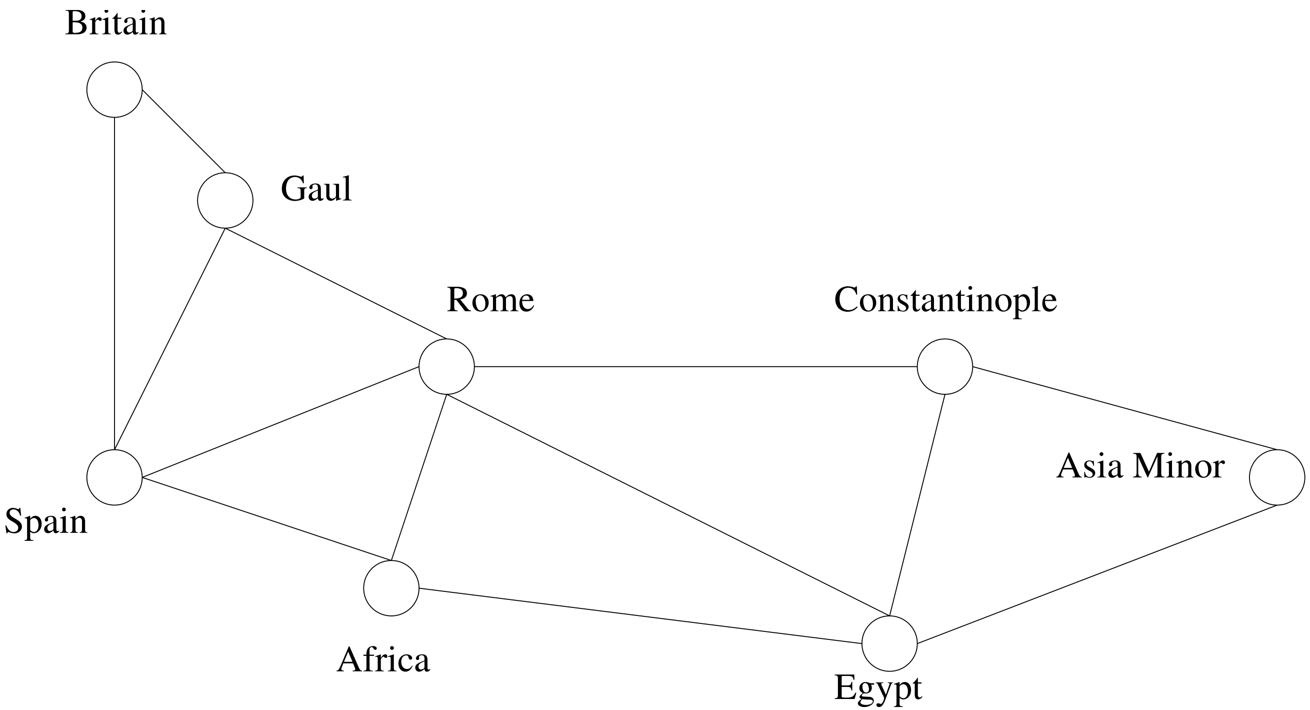
Overview

- Problem definition & introductory example
- $\mathcal{FPT}$: the methodology
- Completeness results
- Algorithmic results

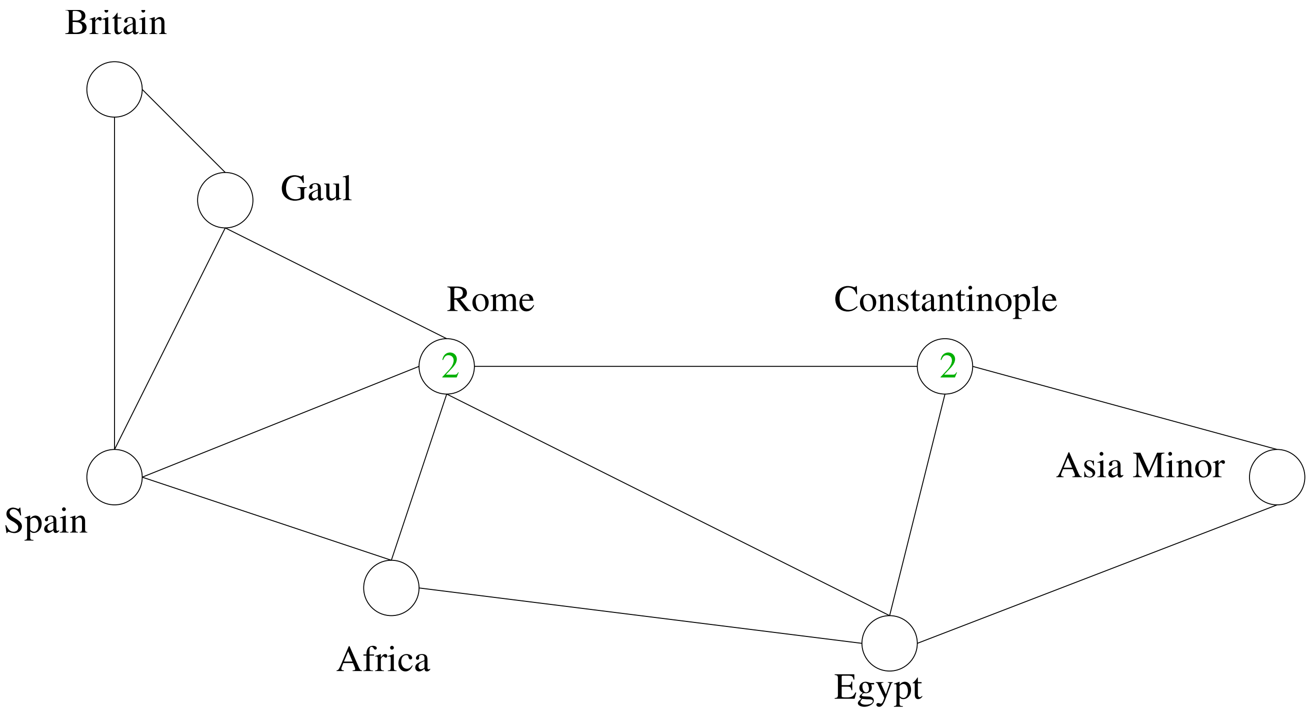
Historical Background The Roman Empire in the times of Constantine



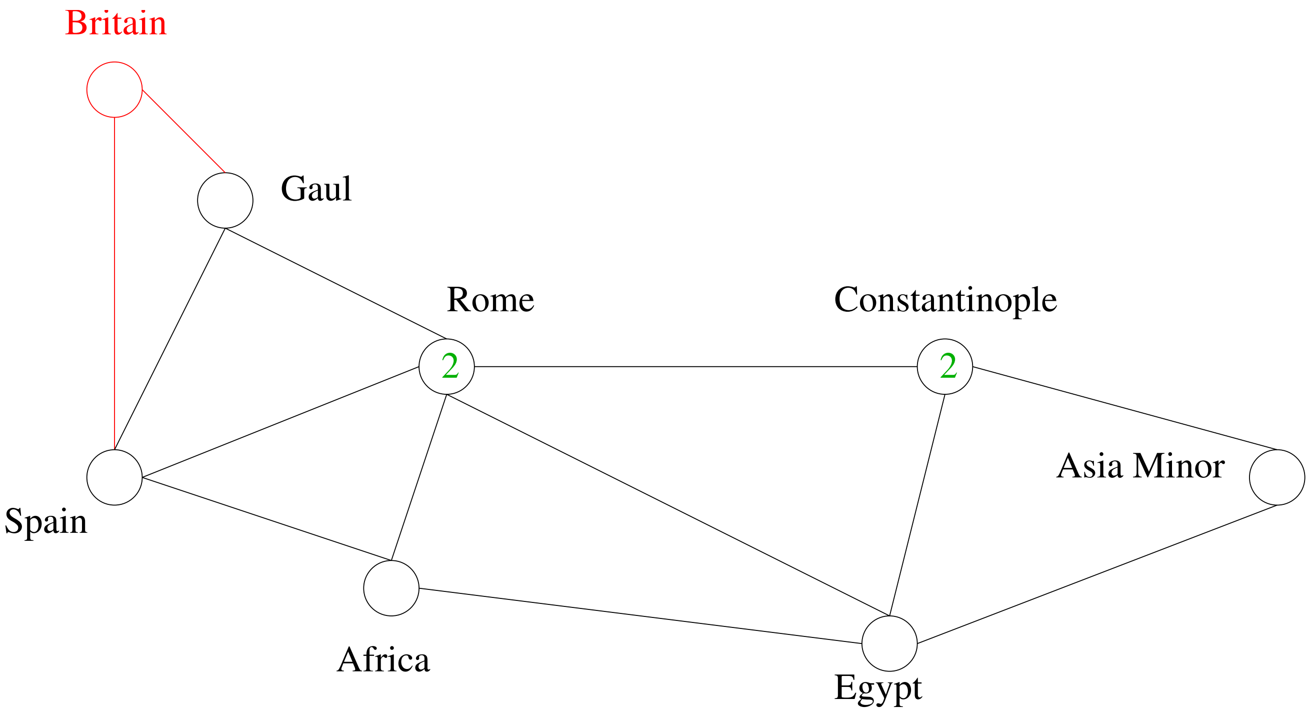
A pure graph model



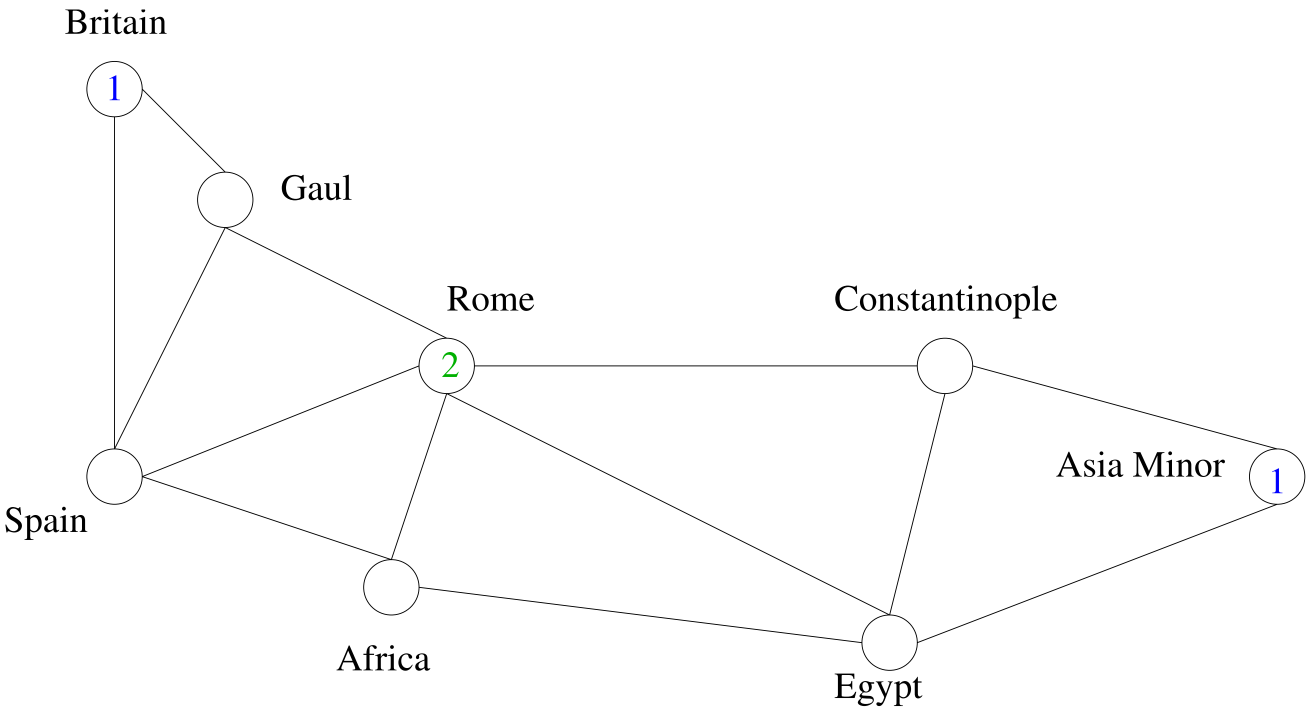
Constantine's solution



Britain in danger



Another solution



Problem definition

A *Roman domination* function of a graph $G = (V, E)$ is a function $R : V \to \{0, 1, 2\}$ with

$$\forall v \in V : R(v) = 0 \Rightarrow \exists x \in N(v) : R(x) = 2.$$

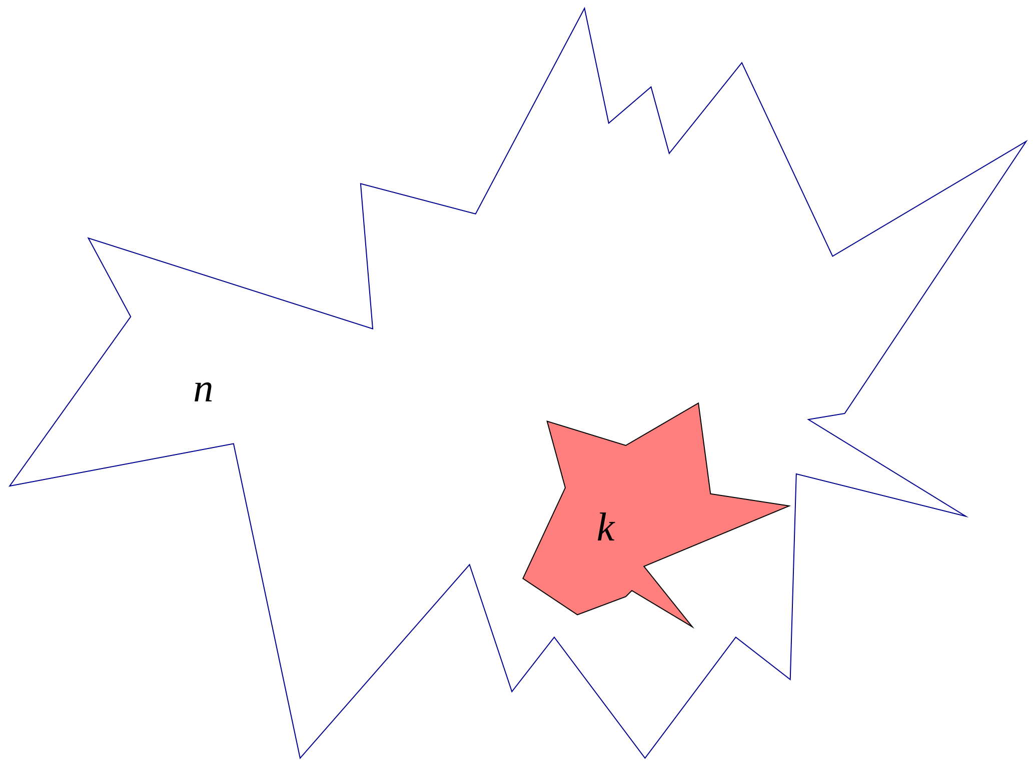
ROMAN DOMINATION (ROMAN)
Given: A graph $G = (V, E)$
Parameter: a positive integer k
Question: Is there a *Roman domination* function R such that

$$R(V) := \sum_{x \in V} R(x) \leq k?$$

Overview

- Problem definition & introductory example
- $\mathcal{FPT}$: the methodology
- Completeness results
- Algorithmic results

The Curse of Combinatorics



Parameterized complexity in a nutshell

Running time $\mathcal{O}(f(k)p(n))$

Problem kernel of size $g(k)$, computable in time $q(n)$.

Thm.: Both approaches yield the same.

Complexity class: $\mathcal{FPT}$

Standard approaches: search trees & kernelization

The hard guys on the Turing way

W[1] can be characterized by the k-step halting problem of single-tape nondeterministic Turing machines.

W[2] can be characterized by the following problem on Turing machines:

SHORT MULTI-TAPE NONDETERMINISTIC TURING MACHINE COMPUTATION
Given: A multi-tape nondeterministic Turing machine M (with two-way infinite tapes), an input string x
Parameter: a positive integer k
Question: Is there an accepting computation of M on input x that reaches a final accepting state in at most k steps?

Parameterized reduction

A *parameterized reduction* is a function r that, for some polynomial p and some function g, is computable in time $\mathcal{O}(g(k)p(|I|))$ and maps an instance (I, k) of $\mathcal{P}$ onto an instance $r(I, k) = (I', k')$ of $\mathcal{P}'$ such that

- (I, k) is a YES-instance of $\mathcal{P}$ if and only if (I', k') is a YES-instance of $\mathcal{P}'$ and
- $k' \leq g(k)$.

We also say that $\mathcal{P}$ *reduces to* $\mathcal{P}'$.

Remark: $\mathcal{FPT} \subseteq$ W[1] $\subseteq$ W[2] $\ldots$

Overview

- Problem definition & introductory example
- $\mathcal{FPT}$: the methodology
- Completeness results
- Algorithmic results

$\boxed{\text{Completeness results}}$

Theorem 1 ROMAN DOMINATION *is W[2]-complete.*

Membership in W[2]

$G = (V, E)$: an instance of ROMAN DOMINATION; let $k > 0$.

The corresponding Turing machine T has $|V| + 1$ tapes; let they be indexed by $\{0\} \cup V$.
Tape symbols: $(V \times \{1, 2\})$ on tape 0 and $\#$ on the other tapes.
The edge relation of G is “hard-wired” into the transition function of T.
The input string of T is empty.

First phase: T nondeterministically guesses the Roman domination function R and writes it on tape 0 using the letters from $V \times \{1, 2\}$ as follows:
T moves the head on tape 0 one step to the right, and writes there a guess $(v, i) \in (V \times \{1, 2\})$.
Upon writing (v, i), T also increments an internal-memory counter c by i.
If $c \leq k$, T can nondeterministically continue in phase one or transition into phase two;
if $c > k$, T hangs up.

Second phase: T has to verify its guess.
Upon reading symbol $(v, 1)$ on tape 0, T writes $\#$ on the tape addressed by v and moves that head one step to the right.
Upon reading $(v, 2)$ on tape 0, T writes $\#$ on all tapes addressed by vertices from $N[v]$ and moves the corresponding heads one step to the right.
Moreover, after reading symbol (v, i) on tape 0, T moves the head on tape 0 one step to the left.
Upon reading the blank symbol on tape 0, T moves all other heads one step to the left;
only if then all V-addressed tapes show $\#$ under their respective heads, T accepts.

Time analysis:

The first phase takes k steps.
The second phase takes another $k + 1$ steps.

Hence, (G, k) is a YES-instance to ROMAN DOMINATION iff T has an accepting computation within $2k + 1$ steps, so that we actually described a parameterized reduction.

Hardness for W[2]

We will show W[2]-hardness with the help of the following problem:

RED-BLUE DOMINATING SET (RBDS)
Given: A graph $G = (V, E)$ with V partitioned as $V_{\text{red}} \uplus V_{\text{blue}}$
Parameter: a positive integer k
Question: Is there a *red-blue dominating set* $D \subseteq V_{\text{red}}$ with $|D| \leq k$, i.e., $V_{\text{blue}} \subseteq N(D)$?

Lemma 2 *(Downey/Fellows)* RED-BLUE DOMINATING SET, RESTRICTED TO BIPARTITE GRAPHS *is W[2]-hard.*

Assume that $G = (V, E)$ is an instance of RED-BLUE DOMINATING SET, RESTRICTED TO BIPARTITE GRAPHS, i.e., $V = V_{\text{red}} \uplus V_{\text{blue}}$. W.l.o.g., $|V_{\text{red}}| > 1$.

In the simulating ROMAN DOMINATION instance, we construct a graph $G' = (V', E')$, where

$$V' = (V_{\text{red}} \cup \{1, \dots, 2k+1\}) \times \{1, \dots, k\} \cup V_{\text{blue}},$$

and E' contains the following edges (and no others):

1. $G'[V_{\text{red}} \times \{i\}]$ is a complete graph for each $i \in \{1, \dots, k\}$.
2. For all $i \in \{1, \dots, k\}$ and $x \in V_{\text{red}}$, $y \in V_{\text{blue}}$, $\{x, y\} \in E$ iff $\{[x, i], y\} \in E'$.
3. For all $i \in \{1, \dots, k\}$, $j \in \{1, \dots, 2k+1\}$ and $x \in V_{\text{red}}$: $\{[x, i], [j, i]\} \in E'$.

<u>Claim</u>: G has a red-blue dominating set D of size k iff G' has a Roman domination function R with $\sum_{x \in V'} R(x) = 2k$.

Overview

- Problem definition & introductory example
- $\mathcal{FPT}$: the methodology
- Completeness results
- Algorithmic results

A search tree result for planar graphs

Theorem 3 PLANAR ROMAN DOMINATION *can be solved in* $\mathcal{O}^*(3.3723^k)$ *time.*

Necessary ingredients:

Adaptation of earlier results on kernelization and search tree algorithms for PLANAR DOMINATING SET.

For the search tree part, a Euler type argument is needed.

Dynamic programming for graphs of bounded treewidth

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

$$\max\{|X_i| \mid i \in I\} - 1.$$

The *treewidth* of G is the minimum k such that G has a tree decomposition of width k, also written $\mathrm{tw}(G)$ for short.

A tree decomposition with a particularly simple structure is given by the following definition.

A tree decomposition $\langle\{X_i \mid i \in I\}, T\rangle$ with a distinguished root node r is called a *nice tree decomposition* if the following conditions are satisfied:

1. Every node of the tree T has at most 2 children.

2. If a node n has two children n' and n'', then $X_n = X_{n'} = X_{n''}$ (in this case n is called a *join node*).

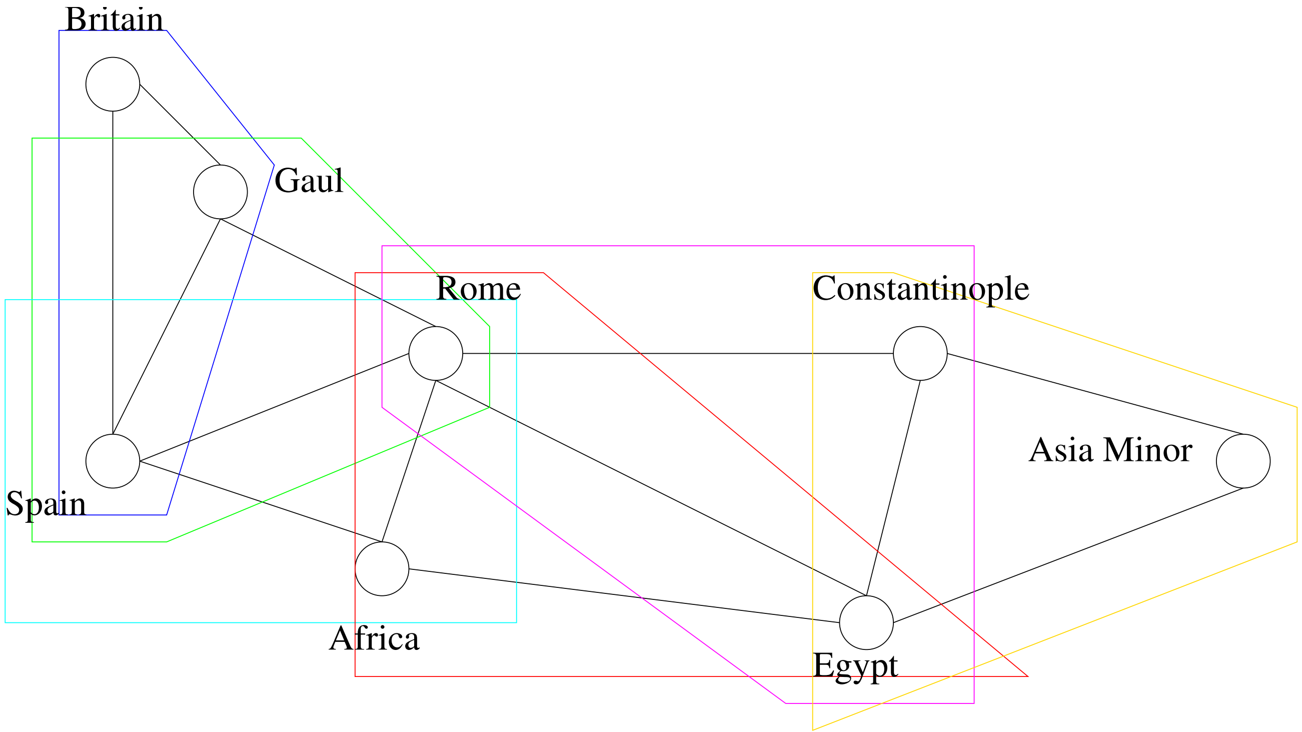
3. If a node n has one child n', then either

   (a) $|X_n| = |X_{n'}| + 1$ and $X_{n'} \subset X_n$ (in this case n is called an *insert node* or an *introduce node*), or

   (b) $|X_n| = |X_{n'}| - 1$ and $X_n \subset X_{n'}$ (in this case n is called a *forget node*).

Observe that each node in a nice tree decomposition is either a join node, an insert node, a forget node, or a leaf node.

Our example revisited (Path decomposition)



Dynamic Programming

We need to store four states per vertex in each node.
0,1,2 are the values that the Roman domination function is assumed to assign to a particular vertex.
$\hat{0}$ also tells us that the Roman domination function assigns 0 to that vertex.
The difference in the semantics of $0, \hat{0}$ is the following:
0: the vertex is already dominated,
$\hat{0}$: we still ask for a domination of this vertex.

Additional complication when dealing with join nodes:
if we update an assignment that maps vertex x onto 0, it is not necessary that both children assign 0 to x; it is sufficient that one of the two branches does, while the other assigns $\hat{0}$.

Alber's monotonicity trick For every vertex x in the parent bag, we consider:

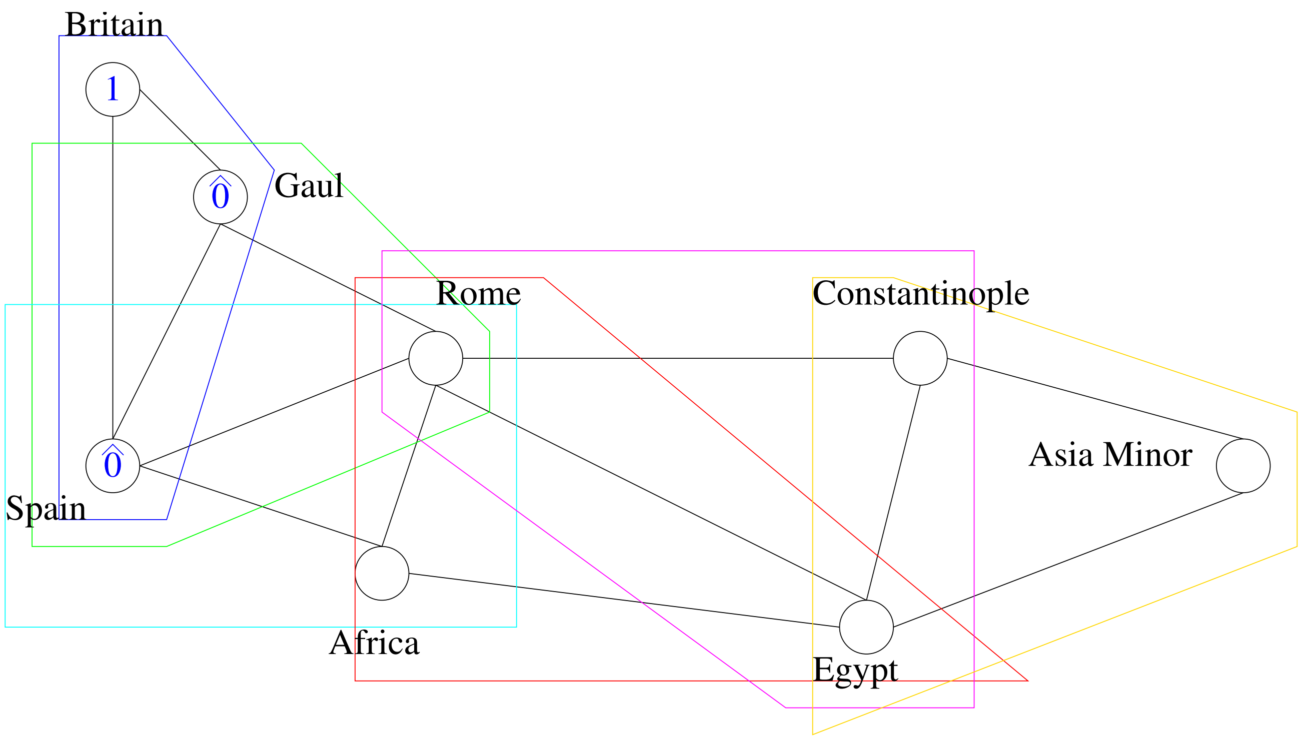
- either 2, 1 or $\widehat{0}$ is assigned to x; then, the same assignment must have been made in the two children;

- or 0 is assigned to x; then, we have two possible assignments in the child nodes: 0 to x in the left child and $\widehat{0}$ to x in the right child or vice versa.

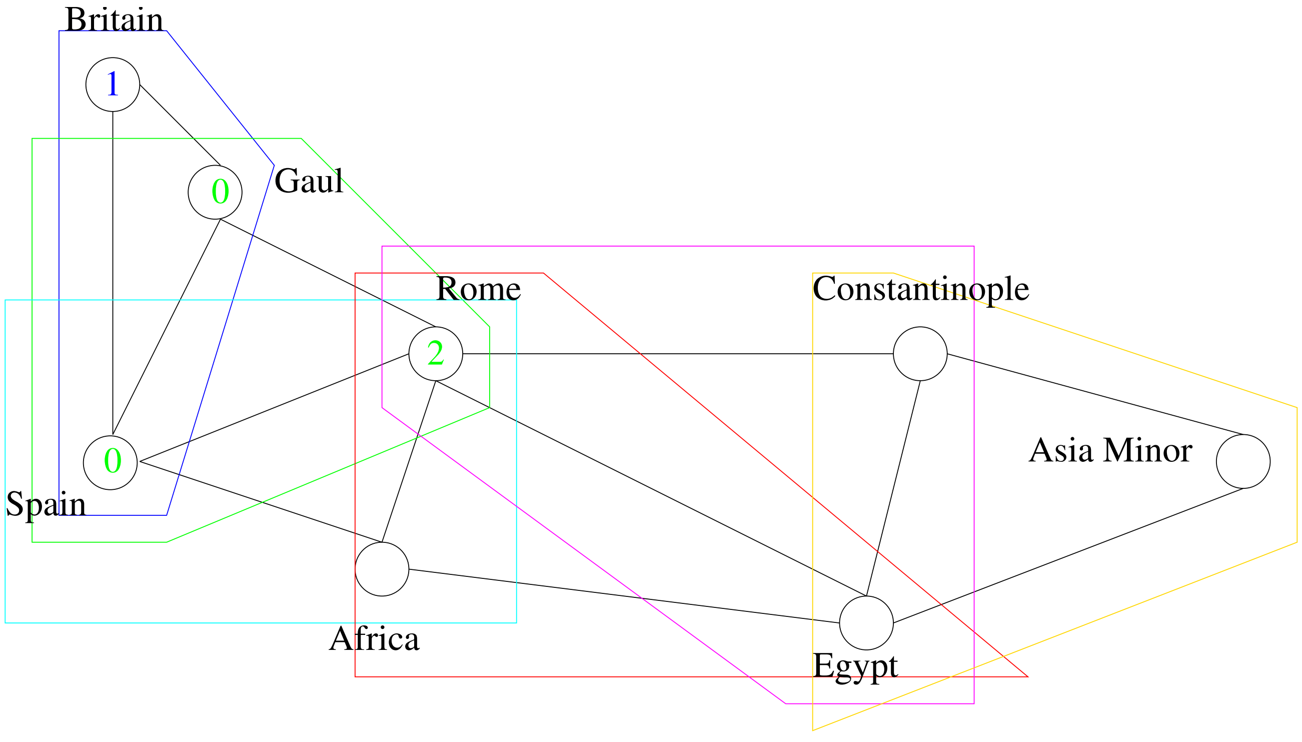
Theorem 4 MINIMUM ROMAN DOMINATION, *parameterized by the treewidth* $\mathrm{tw}(G)$ *of the input graph* G, *can be solved in time* $\mathcal{O}(5^{\mathrm{tw}(G)}|V(G)|)$.

Remark: Complexity $\mathcal{O}(4^{\ell}|V(G)|)$ if ℓ is the pathwidth of G.

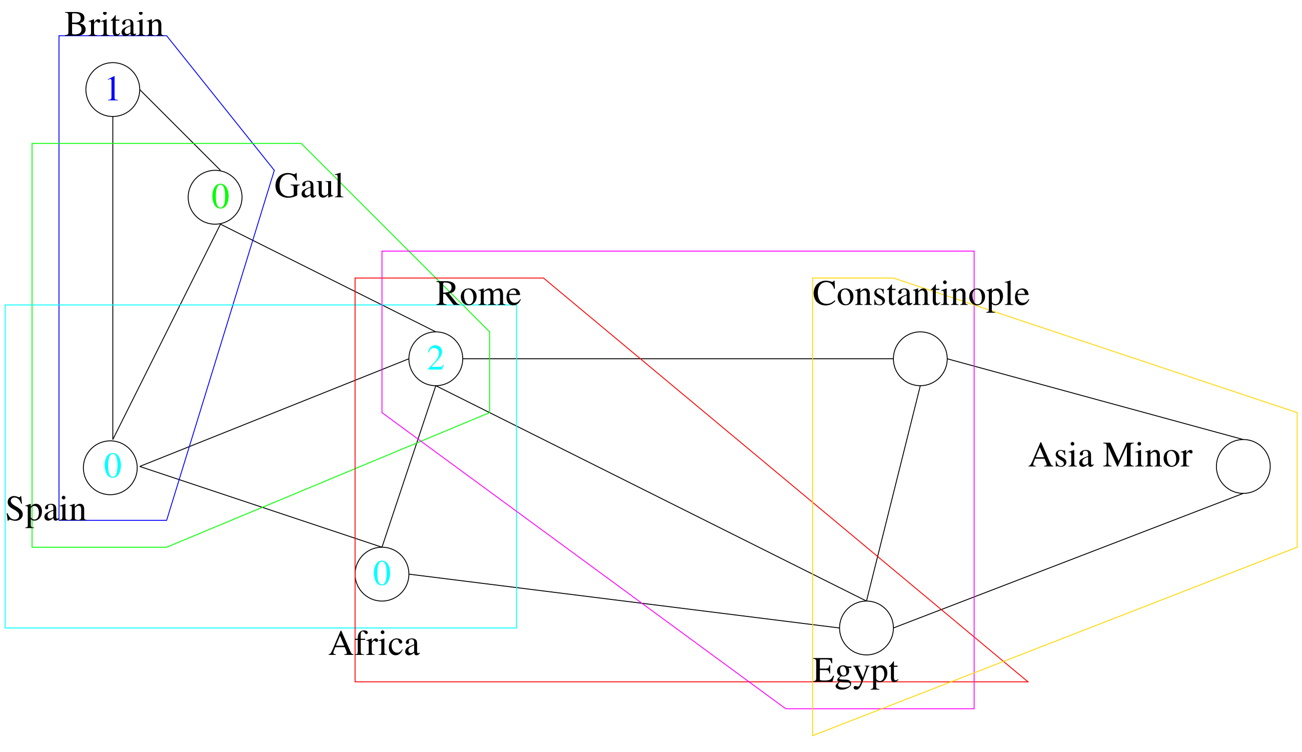
Our example revisited (Path decomposition, 1st bag)



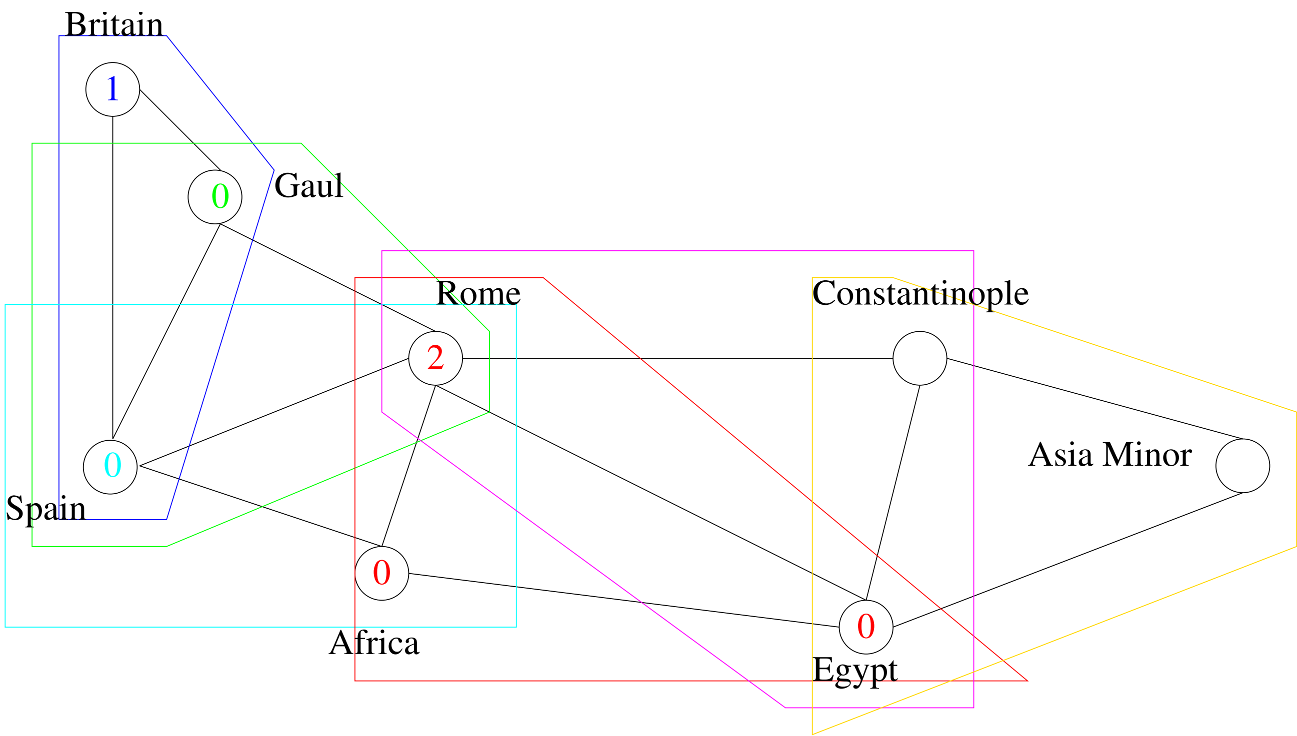
Our example revisited (Path decomposition, 2nd bag)



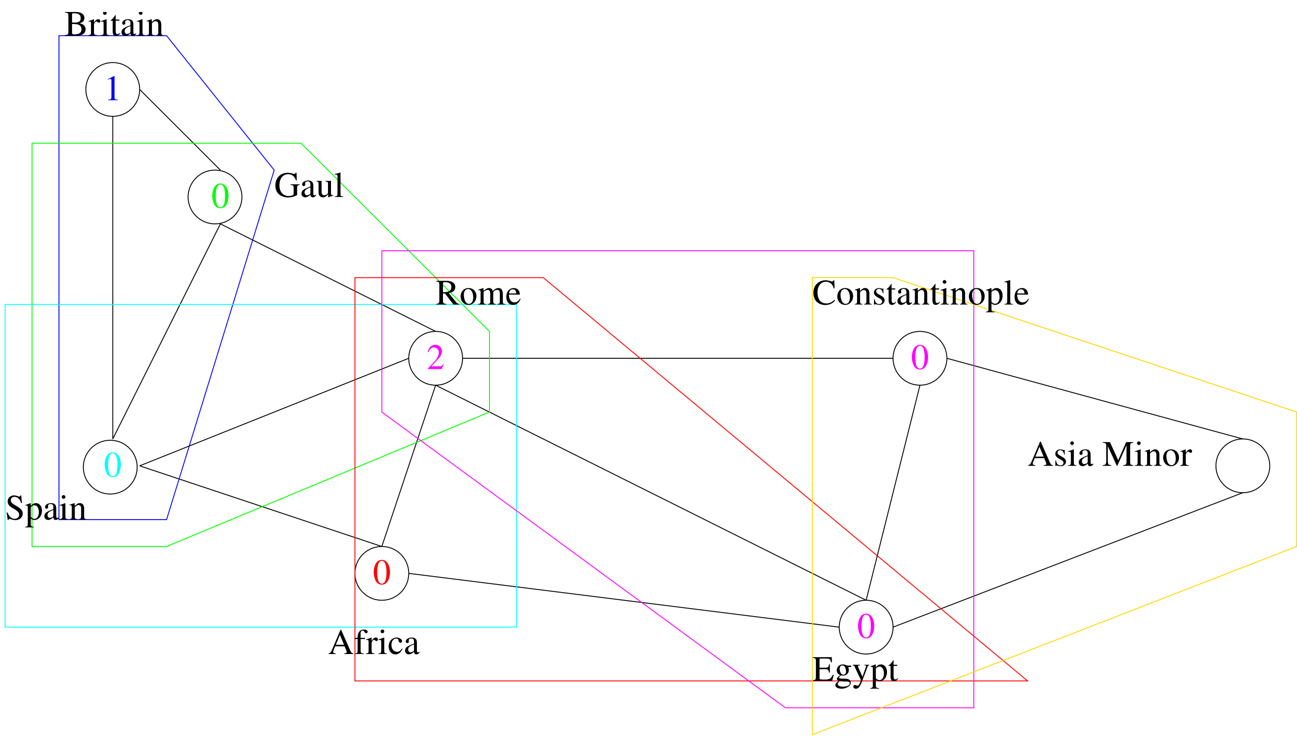
Our example revisited (Path decomposition, 3rd bag)



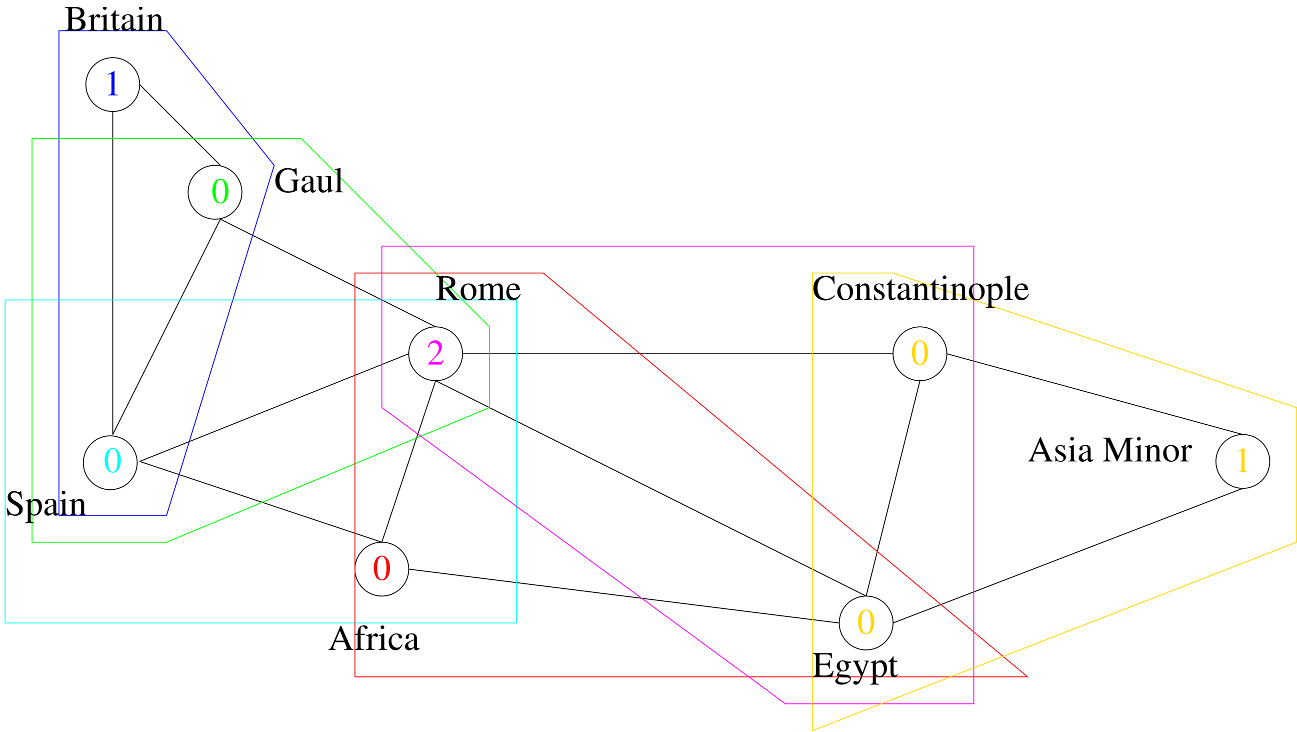
Our example revisited (Path decomposition, 4th bag)



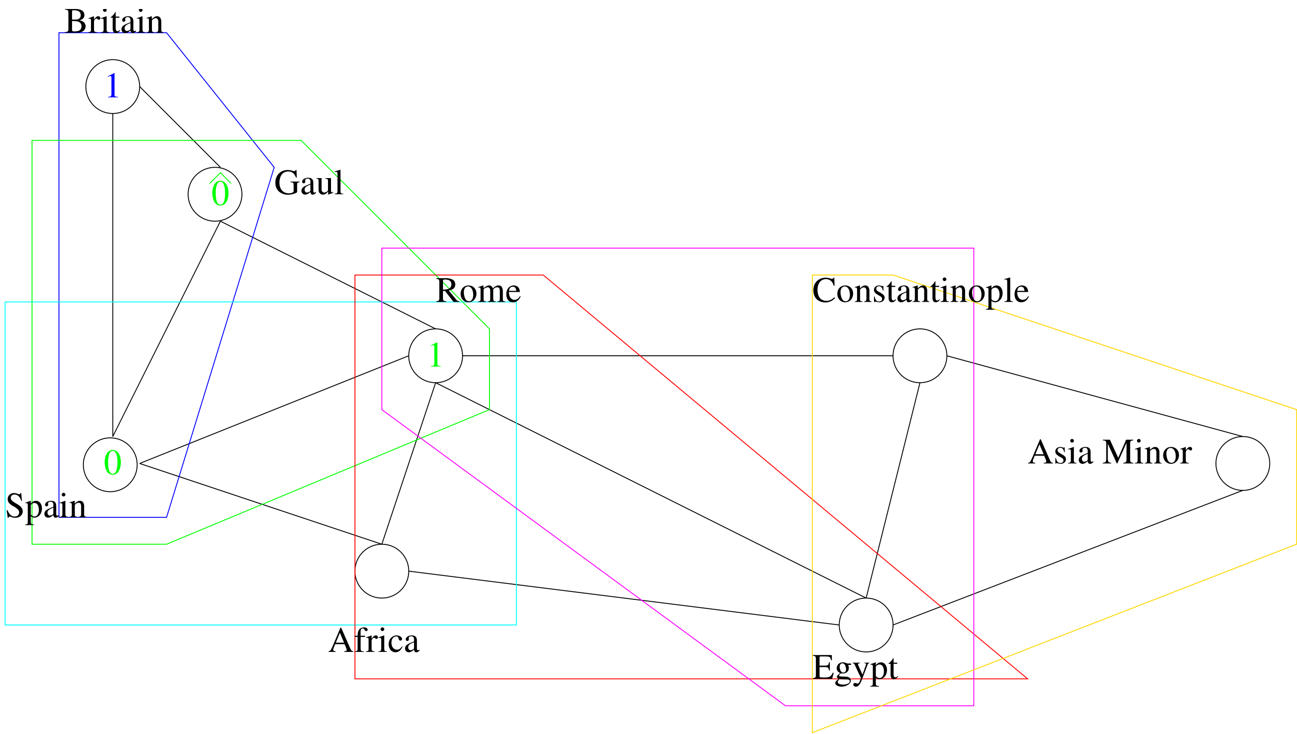
Our example revisited (Path decomposition, 5th bag)



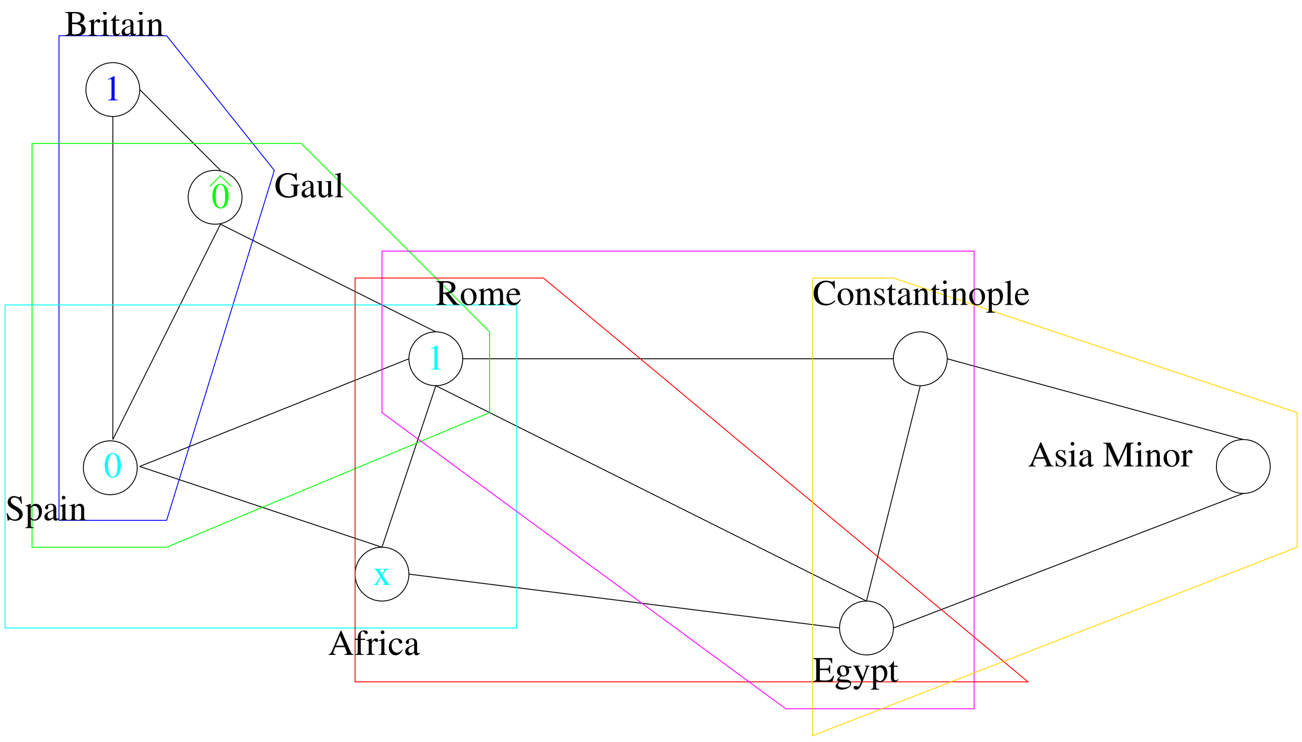
Our example revisited (Path decomposition, 6th bag)



Our example revisited (Path decomposition, 2nd bag, bad guess)



Our example revisited (Path decomposition, 3rd bag, bad guess)



Application to planar graphs

Theorem 5 *[Fomin and Thilikos 2003] If G is a planar graph which has a dominating set of size k, then G has treewidth of at most $4.5^{1.5}\sqrt{k} \leq 9.55\sqrt{k}$.*

Corollary 6 PLANAR ROMAN DOMINATION *can be solved in time*

$$\mathcal{O}^*(5^{4.5^{1.5}\sqrt{k}}) = \mathcal{O}^*(2^{22.165\sqrt{k}}).$$

A dual version of ROMAN DOMINATION

We finally mention that the following version of a parametric dual of ROMAN is in $\mathcal{FPT}$ by the method of kernelization, given a graph G and a parameter k_d, is there a Roman domination function R such that $|R^{-1}(1)| + 2|R^{-1}(0)| \geq k_d$?

With our definition, we have the desirable property that (G, k_d) is a YES-instance of this variant of a dual of ROMAN DOMINATION iff $(G, 2|V(G)| - k_d)$ is a YES-instance of ROMAN. In other words, R is maximum for this dual version of ROMAN iff R is minimum for ROMAN.

Theorem 7 *Our version of parametric dual of* ROMAN DOMINATION *allows for a problem kernel of size $(7/6)k_d$, measured in terms of vertices. Hence, this problem is in $\mathcal{FPT}$.*

Take Away

- As can be seen from the W[2] completeness section, the “Turing way” to parameterized complexity is often quite amenable and may offer advantages over the standard approach as exhibited in [DowFel99].

- Strive to obtain structural results when developing algorithms: this turned out to be very beneficial for PLANAR ROMAN DOMINATION, since the results obtained for PLANAR DOMINATING SET could be “recycled.”