Implementing the Tractable-Design Cycle: Definitions and Techniques

Ulrike Stege
Department of Computer Science
University of Victoria
Implementing the Tractable-Design Cycle: Complexity Analysis—Definitions and Techniques

Joint work with Iris van Rooij, cf. Iris’ PhD thesis.
Outline

- Motivation

- Computational problems
  - Optimization Problems versus Decision Problems

- Classical Tractability vs. Classical Intractability
  - Classical Tractability and Polynomial time
  - Nondeterministic polynomial time
  - Classical Intractability: NP-hardness and NP-completeness

- Parameterized decision problems
  - Fixed-Parameter Tractability
  - Fixed-Parameter Intractability
Which functions can describe cognitive systems?
Which functions can describe cognitive systems?

- All functions
- Turing-computable functions
- Cognitive functions

intractable
tractable
Which cognitive systems are tractable?

- Before Testing of Cognitive Theory
  - Formalization of the cognitive theory → computational problem

- Tractable or intractable? → Analyze complexity of computational problem
- If intractable, revise cognitive theory
Computational Problems

- Optimization Problems
- Decision Problems
- Optimization Problems versus Decision Problems
Optimization Problems

- Maximization problems
- Minimization problems

Coherence (informal)

Input: A set of interconnected beliefs.
Output: A truth assignment of maximum coherence.
Coherence (informal)
Input: A set of interconnected beliefs.
Output: A truth assignment of maximum coherence.

Coherence (formal)
Input: Set of propositions $P$, set of constraints $C = C^- \cup C^+$.

Output: A truth assignment to the propositions in $P$ that satisfies a maximum number of constraints. Here a constraint $(p, q) \in C^-$ is satisfied if $p$ is ‘false’ and $q$ is ‘true’, and a constraint $(p, q) \in C^+$ is satisfied if both $p$ and $q$ are ‘true’ or both $p$ and $q$ are ‘false’.
Optimization Problems

Coherence (even more formal)

**Input:** Network $N = (P,C)$, where $C$ is partitioned into $C = C_- \cup C_+$.  

**Output:** A subset $A \subseteq P$ such that $|\{(p,q) \in C : (p,q) \text{ is satisfied}\}|$ is maximized.

Here, $(p,q) \in C_-$ is satisfied if either $(p \in A$ and $q \notin A)$ or $(p \notin A$ and $q \in A)$, and $(p,q) \in C_+$ is satisfied if either $p, q \in A$ or $p, q \notin A$. 
Computational Problems

- Optimization Problems
- Decision Problems
- Optimization Problems versus Decision Problems
Decision Problems

- Answer: yes / no
Decision Problem

**Coherence** (Decision Version)

*Input:* \( N = (P, C) \), \( C \) is partitioned into \( C = C^- \cup C^+ \), a positive integer \( k \)

*Question:* Does there exist \( A \subseteq P \) such that \( |\{(p,q) \in C : (p,q) \text{ is satisfied}\}| \geq k \)?
Decision Problems

- Answer: yes / no
- Often answer is **constructive**: If yes, we also know a solution that is a **witness** for answer.
Computational Problems

- Optimization Problems
- Decision Problems
- Optimization Problems versus Decision Problems
Decision Problems versus Optimization Problems

- **Goal**: Determine whether the formalized (optimization) problem is tractable or intractable.
- **Complexity Theory**: set up for decision problems
- **What about our optimization problem??**
Decision Problems versus Optimization Problems

- In classical and parameterized framework we can show

  If decision problem is tractable, then optimization problem is tractable, and vice versa!

Why?
We first introduce our framework and then reconsider this issue.

**Classical Complexity**

- Tractability $\equiv$ Polynomial Time $\equiv \mathcal{P}$

- A decision problem $L$ is *decidable in polynomial time* iff for each instance $<x,k>$ it can be decided in $|x|^c$ ($c$ is constant) time whether $<x,k> \in L$ or $<x,k> \notin L$. 
### The Polynomial Time Class \( P \)

- **Examples:** \(|x|, |x|^2, |x|^3, |x|^{81}\)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( O(n^2) )</th>
<th>( O(2^n) )</th>
<th>( O(2^{\kappa n}), \kappa = 10 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.15 msec</td>
<td>0.19 msec</td>
<td>0.51 sec</td>
</tr>
<tr>
<td>20</td>
<td>0.04 sec</td>
<td>1.75 min</td>
<td>2.05 sec</td>
</tr>
<tr>
<td>50</td>
<td>0.25 sec</td>
<td>8.4 x 10^3 yrs</td>
<td>5.12 sec</td>
</tr>
<tr>
<td>100</td>
<td>1.00 sec</td>
<td>9.4 x 10^{17} yrs</td>
<td>10.2 sec</td>
</tr>
<tr>
<td>1000</td>
<td>1.67 min</td>
<td>7.9 x 10^{288} yrs</td>
<td>1.71 min</td>
</tr>
</tbody>
</table>
How can we prove that a decision problem \( L \in \mathcal{P} \)?
Decision Problems versus Optimization Problems

- In classical and parameterized framework we can show

If optimization problem is tractable, then decision problem is tractable.

\[ L_{\text{opt}} \in \mathcal{P} \Rightarrow L \in \mathcal{P} \]
$L_{\text{opt}} \in \mathcal{P} \Rightarrow L \in \mathcal{P}$
Decision Problems versus Optimization Problems

- In classical and parameterized framework we can show

If decision problem is tractable, then optimization problem is tractable!

Why?
Coherence for $C = C^+$

$k = 6$

$P' = ?$
Coherence for $C = C^+$ (opt. Version)

$P' = ?$
\[ L \in \mathcal{P} \Rightarrow L_{\text{opt}} \in \mathcal{P} \]

- \( x \in L_{\text{opt}}? \)
- Range of \( k \)
- \( \langle x, k \rangle \)
- (Range of \( k \)) |\!| x |\!| ^c \in \mathcal{P} \]
Decision Problems versus Optimization Problems

\[ L \in \mathcal{P} \iff L_{\text{opt}} \in \mathcal{P} \]
Coherence (Decision Version)

Input: $N = (P, C)$, $C$ is partitioned into 
$C = C_- \cup C_+ $, a positive integer $k$

Question: Does there exist $A \subseteq P$ such that $|\{(p, q) \in C : (p, q) \text{ is satisfied}\}| \geq k$?

- Is Coherence $\in \mathcal{P}$?
- Answer: we don’t really know.
- What do we know?
The Polynomial Time Class $\mathcal{P}$

- How can we prove that a decision problem $L \in \mathcal{P}$?
- How can we prove that a decision problem $L \notin \mathcal{P}$?
A decision problem $L$ is *decidable in nondeterministic polynomial time* iff for each instance $<x,k>$ and any solution $S$, it can be verified in polynomial time ($|x|^c$) if $S$ proves that $<x,k>$ is a yes-instance for $L$.
The Nondeterministic Polynomial Time Class $\textit{NP}$

- For a decision problem $L \in \textit{NP}$ and an instance $<x,k>$ we can determine whether $<x,k>$ is a yes-instance for $L$ or $<x,k>$ is a no-instance for $L$ in \textit{exponential time}.

- That is, we just have to try out each candidate solution!
Coherence $\in \text{NP}$

- What is a possible solution/witness for Coherence?

- Show
  - The witness is "short"

  - We can verify in a "short" time if the witness is a correct solution.
A task that we can complete fast, we can also complete slow(er).

Thus: $P \subseteq NP$

But: the converse does not necessarily hold!
$P = NP$?

- Million dollar question!
- Assumption: $P \neq NP$
- We assume there is a decision problem $L$ such that $L \in P$ and $L \notin NP$.
- We say a problem is $NP$–hard if it is at least as hard as any problem in $NP$.
- We say a problem is $NP$–complete if it is
  (1) $NP$-hard and (2) also in $NP$ itself.
(Classically) intractable

- An $NP$-hard decision problem is viewed as (classically) intractable.
- To prove that $P = NP$ it is enough to show that there exists an $NP$-hard problem that is in $P$!
- So far, nobody was able to do so …
Proving \( 	ext{NP} \)-hardness

Let \( L \) be the problem we want to show \( \text{NP} \)-hardness for.

- Show that there is an \( \text{NP} \)-hard problem \( L' \) that can be polynomial-time reduced to \( L \).

\[
\begin{align*}
<x', k'> &\in L'？ \\
\text{Polynomial-time algorithm} &\rightarrow \\
p(<x', k'>) &\in L？
\end{align*}
\]

yes \quad \text{if and only if} \quad \text{yes}
Proving $\mathcal{NP}$–hardness

If $L$ is $\mathcal{NP}$–hard, then a polynomial-time algorithm for $L$ would also imply a polynomial-time algorithm for $L'$.

How do we find $L'$?

- There is a huge catalogue of problems that are shown to be $\mathcal{NP}$–hard, just pick one that works without too much trouble.
- Not very difficult, but experience helps.
Coherence is \( \mathcal{NP} \)-hard

- We reduce from a problem called Max-Cut
- Max-Cut is known to be \( \mathcal{NP} \)-complete [GJ’79].
- More details in demo session tomorrow.
Corollary

- Coherence is $\mathcal{NP}$-complete
  – even for $C = C^-$
Is Coherence (really) intractable?

Reconsider / specify the task.
- Can we reduce the input space?
- Can we parameterize?
Reducing the input space: $C = C^+$

- We can decide Coherence in polynomial time if $C = C^+$. 
Reducing the input space: Is a network consistent?

- A network is **consistent** if every edge (constrained) can be satisfied.

- We can decide in polynomial time whether or not a network $N = (P, C)$ is consistent.
Reducing the input space: Coherence for trees
Reducing the input space: Coherence for trees
Reducing the input space:
Coherence for trees
Reducing the input space: Coherence for trees
Reducing the input space: Coherence for trees
Reducing the input space:
Coherence for trees
Reducing the input space:
Coherence for trees
Reducing the input space: Coherence for trees

Every tree is a consistent network!
Is Coherence (really) intractable?

Reconsider / specify the task.

- Can we reduce the input space?
- Can we parameterize?
Parameterized Complexity

- Parameterized decision problem
- Parameterized Complexity Classes
  - \( FPT \)
  - \( W[1] \)
  - \( W[2] \)
  - ...
Like decision problem, but a parameter (explicit or implicit) is specified.
**k-Coherence (Parameterized Decision Version)**

**Input:** An (inconsistent) network $N = (P,C)$, $C$ is partitioned into $C = C_- \cup C_+$, a positive integer $k$

**Parameter:** $k$

**Question:** Does there exist $A \subseteq P$ such that $|\{(p,q) \in C : (p,q) \text{ is satisfied}\}| \geq k$?
A parameterized decision problem $L$ is fixed-parameter tractable ($fpt$) if there exists a constant $\alpha$ and an algorithm $\Phi$ such that $\Phi$ decides if $\langle x, k \rangle$ is a yes-instance for $L$ in time $f(k) \cdot |x|^\alpha$ where $f$ is an arbitrary function of the parameter $k$. 
Examples for \textit{FPT} running times

instance size $|x| = n$, parameter $k$

$2^k n$

$2817^{2k} + n^3$

$n^{91}$

$k^{k^k} + n$
Remarks

A problem that is $NP$–hard or $NP$–complete for can be fixed-parameter tractable for a chosen parameter!

A problem that is in $P$ is fixed-parameter tractable for any chosen parameter.
**k-Coherence**

*Input:* An (inconsistent) network \( N = (P,C) \), \( C \) is partitioned into \( C = C^- \cup C^+ \), a positive integer \( k \)

*Parameter:* \( k \)

*Question:* Does there exist \( A \subseteq P \) such that \(|\{(p,q) \in C : (p,q) \text{ is satisfied}\}| \geq k\)?
$k$-Coherence for connected networks is in $\text{FPT}$

- A network $N$ is connected if for every pair of nodes there exists a path in $N$. 
We have that $k$-Coherence for connected networks is in \( \mathcal{FPT} \).

**Lemma.** Let \(<N, k>\) be an instance for \(k\)-Coherence, \(N\) connected. If \(|P| > k\), then \(<N, k>\) is a yes-instance.

**Proof.**
**Lemma.** Let $<N, k>$ be an instance for $k$-Coherence, $N$ connected. If $|P| > k$, then $<N, k>$ is a yes-instance.

**Proof.** Pick a spanning subtree of $N$. 
Lemma. Let $<N, k>$ be an instance for $k$-Coherence, $N$ connected. If $|P| > k$, then $<N, k>$ is a yes-instance.

Proof. Pick a spanning subtree of $N$. \Rightarrow \text{At least } |P| \text{ edges are satisfied!}
$k$-Coherence for connected networks is in $\text{FPT}$

- $\langle N, k \rangle$ instance for $k$-Coherence
- $N$ connected
- If $|P| > k$ then $\langle N, k \rangle$ is a yes-instance.
- Else $|P| \leq k$.
- How much time does it take to decide the answer for $\langle N, k \rangle$ with $|P| \leq k$?
- $|E| = ?$
**k-Coherence for connected networks is in \( \text{FPT} \)**

- \(<N, k>\) instance for \( k \)-Coherence
- \( N \) connected
- If \(|P| > k\) then \(<N, k>\) is a yes-instance.
- Else \(|P| \leq k\).
- How much time does it take to decide the answer for \(<N, k>\) with \(|P| \leq k\)?
- \(|E| \leq \binom{k}{2}\)
$k$-Coherence for connected networks is in $\textit{FPT}$

- $N$ is bounded in size in a function of $k$
  - Problem Kernel
- Even if we have to try out every possible solution, the number of those is still a function of $k$
- We can answer in fixed-parameter-tractable time for parameter $k$
Proving \textbf{\textit{FPT}}-Membership

- Give an algorithm
  - Problem Kernel / Kernelization
  - Bounded Search Tree
- Prove existence of a problem kernel
  - Boundary Lemma
- Graph Minor Theorem
- More details in demo session tomorrow.
How do we show that a (parameterized decision) problem is parameterized intractable?

- Prove that the problem is hard for class $\mathcal{W}[1]$ or class $\mathcal{W}[2]$ or ...
- Prove that: if the problem is in $FPT$, then $\mathcal{P} = \mathcal{NP}$. 
$W[1]$

- $FPT \subseteq W[1]$
- Conjecture: $FPT \neq W[1]$
- Hard for $W[1]$
  - Problems that are likely not fixed-parameter tractable
  - Running times are something like $n^k$
Prove that a (parameterized) problem is hard for class $\mathcal{W}[1]$

- Via parameterized reduction from a problem that is known to be hard for $\mathcal{W}[1]$ and that further preserves the parameter.

- Similar idea as in $\mathcal{NP}$–hardness proofs.
  - 😊 Time permitted is in $\mathcal{FPT}$ (any function in parameter is allowed, rest polynomial)
  - 😞 Parameter has to be preserved!

😊 Many of the “classic” $\mathcal{NP}$–hardness reductions in the literature are already parameterized.
We investigated techniques from computer science to prove (in)tractability for decision problems and optimization problems.

We also observed: If a special case of a decision problem is \( NP \)-hard, then the problem itself is \( NP \)-hard itself.

Further: If we can prove that a problem is tractable, then its special cases are tractable as well.
Demo Session

Ulrike Stege (University of Victoria)
Iris van Rooij (TU Eindhoven)
Topics

- $NP$–completeness proofs
  - Membership
  - Polynomial-time reduction

- $FPT$–algorithms
  - Technique of building a problem kernel
  - Technique of bounded a search tree
  - combination
Topics

- **FPT** – membership
  - Existence of a problem kernel
  - Graph Minor Theorem
- Parameterized intractability
  - Parameterized reduction
  - Not in $\text{FPT}$ unless $P = NP$
  - Membership in $W[1]$
Coherence (Decision Version)

Input: An (inconsistent) network \( N = (P,C) \),
\( C \) is partitioned into \( C = C_- \cup C_+ \), a positive integer \( k \)

Question: Does there exist \( A \subseteq P \) such that \(|\{(p,q) \in C : (p,q) \text{ is satisfied}\}| \geq k|\)?
Coherence is $\mathcal{NP}$–hard

We reduce from

**Max-Cut** (decision version)

*Input*: A graph $G = (V, E)$. A positive integer $m$.

*Question*: Does there exist a partition of $V$ into sets $A$ and $R$ such that $\left| \{(u, v) \in E : u \in A, v \in R\} \right| \geq m$?

- Max-Cut is known to be $\mathcal{NP}$–complete [GJ’79].
Max Cut (Example)

\[ m = 4 \]
Let $L$ be the problem we want to show $NP$–hardness for.

- Show that there is an $NP$–hard problem $L'$ that can be polynomial-time reduced to $L$.

\[
\begin{array}{ccc}
<x', k'> \in L' \? & \text{Polynomial-time algorithm} & p(<x', k'>) \in L \? \\
\text{yes} & \text{if and only if} & \text{yes}
\end{array}
\]
Coherence is \( \mathcal{NP} \)-hard

- Let \((G,m)\) be an instance for Max Cut.
- We define an instance \(<N,k>\) for Coherence as follows.
  - \(P = V\)
  - \(C = E\)
  - \(C^{-} = E\)
  - \(k = m\)
- We still have to prove

\(<G,m>\) is a yes-instance for Max-Cut if and only if \(<N,k>\) is a yes-instance for Coherence
<G,m> is a yes-instance for Max-Cut if and only if <N,k> is a yes-instance for Coherence.

“⇒”

- Since <G,m> is a yes-instance for Max-Cut, we can assume V be partitioned into A and R. Further let \(|\{(u, v) \in E : u \in A, v \in R\}| \geq m\).
- We show A is a solution for N. Consider an edge \(e \in \{(u, v) \in E : u \in A, v \in R\}\). Edge e is satisfied!
- There are at least \(m = k\) many of those edges!
(G,m) is a yes-instance for Max-Cut if and only if (N,k) is a yes-instance for Coherence.

“⇐”

- Let P’ be a solution for N.
- We define a partition A = P’, R=V-P’ for G. Let e be satisfied in N. Then
  \[ e \in \{(u,v) \in E : u \in A, v \in R\} \text{. Then } \left| \{(u,v) \in E : u \in A, v \in R\} \right| \geq p = m \]
Corollary

- Coherence is $NP$–complete
  - even for $C = C^-$
To determine whether a network is consistent is in $P$. 
Coherence for trees

Every tree is a consistent network!
To determine whether a network is consistent is in $P$. 
Technique of Problem Kernel
**$k$-Coherence (Parameterized Decision Version)**

*Input:* An (inconsistent) network $N = (P,C)$, $C$ is partitioned into $C = C_– \cup C_+$, a positive integer $k$

*Parameter:* $k$

*Question:* Does there exist $P' \subseteq P$ such that $|\{(p,q) \in C : (p,q) \text{ is satisfied}\}| \geq k$?
Lemma. Let \(<N, k>\) be an instance for \(k\)-Coherence, \(N\) connected. If \(|P| > k\), then \(<N, k>\) is a yes-instance.

Proof.
Lemma. Let $\langle N, k \rangle$ be an instance for $k$-Coherence, $N$ connected. If $|P| > k$, then $\langle N, k \rangle$ is a yes-instance.

Proof. Pick a spanning subtree of $N$. 
**Lemma.** Let \(<N, k>\) be an instance for \(k\)-Coherence, \(N\) connected. If \(|P| > k\), then \(<N, k>\) is a yes-instance.

**Proof.**

Pick a spanning subtree of \(N\). ⇒ At least \(|P|\) edges are satisfied!
**k-Coherence for connected networks is in \( \mathcal{FPT} \)**

- \(<N, k>\) instance for \(k\)-Coherence
- \(N\) connected
- If \(|P| > k\) then \(<N, k>\) is a yes-instance.
- Else \(|P| \leq k\).
- How much time does it take to decide the answer for \(<N, k>\) with \(|P| \leq k\)?
- \(|E| = ?\)
\( k \)-Coherence for connected networks is in \( \mathcal{FPT} \)

- \( <N, k> \) instance for \( k \)-Coherence
- \( N \) connected
- If \( |P| > k \) then \( <N, k> \) is a yes-instance.
- Else \( |P| \leq k \).
- How much time does it take to decide the answer for \( <N, k> \) with \( |P| \leq k \)?
- \( |E| \leq \binom{k}{2} \)
$k$-Coherence for connected networks is in $\text{FPT}$

- $N$ is bounded in size in a function of $k$
  - $N$ is a problem Kernel
- Even if we have to try out every possible solution, the number of those is still a function of $k$
- We can answer in fixed-parameter-tractable time for parameter $k$
$k$-Coherence for connected networks is in $\text{FPT}$

- $N$ is bounded in size in a function of $k$
  - $N$ is a problem Kernel

- However: Often this is just the 1$^{\text{st}}$ step of an $fpt$-algorithm.
We show using this technique that the problem $|C|-\text{Coherence}$ is in $\mathsf{FPT}$. 
\textbf{|C−|-Coherence (Parameterized Decision Version)}

\textit{Input:} An (inconsistent) network $N = (P,C)$, $C$ is partitioned into $C = C− \cup C+$, a positive integer $k$

\textit{Parameter:} $|C−|

\textit{Question:} Does there exist $P’ \subseteq P$ such that $|\{(p,q) \in C : (p,q) \text{ is satisfied}\}| \geq k$?
Technique of bounded search
trees – smart exhaustive search
|C−|-Coherence

**Input:** An (inconsistent) network \( N = (P,C) \), 
\( C \) is partitioned into \( C = C− \cup C+ \), a positive integer \( k \)

**Parameter:** \(|C−|\)

**Question:** Does there exist \( P' \subseteq P \) such that \(|\{(p,q) \in C : (p,q) \text{ is satisfied}\}| \geq k\)?
\(|C|\)-Coherence is in \(FPT\)

- Generalize \(|C|\)-Coherence to \(|C|\)-Annotated Coherence
- Apply technique of bounded search tree to \(|C|\)-Annotated Coherence
Generalization of $|C^-|$-Coherence

$|C^-|$-Annotated Coherence

**Input:** A network $N = (P, C)$. Here, $P$ is partitioned into $U^*$, $P^*$, and $R^*$, and $C$ is partitioned into $C^+$ and $C^-$. A positive integer $k$.

**Parameter:** $|C^-|

**Question:** Does there exist a partition of $P$ into $P'$ and $R$ such that $P^* \subseteq P'$, $R' \subseteq R$, and at least $k$ edges are satisfied by $A$ and $R$?
Coherence is a special case of Annotated Coherence.
An $FPT$-Algorithm for $|C|-\text{-Annotated Coherence}$
An \textit{FPT}-Algorithm for $|C|-\text{-Annotated Coherence}$

$P = P_- \cup P_+$
An $FPT$-Algorithm for $|C−|$

Annotated Coherence

$\langle N, k \rangle$

$\langle N_1, k_1 \rangle$

Select $p$

$\langle N_2, k_2 \rangle$

Do not select $p$

$p \in P$
\((P-)\)-Element-decision Branching-Rule

\(<N, k>\): instance for \(|C|-\)Annotated Coherence

- \(N = (P, C)\)
- \(P = U' \cup P' \cup R'\), and
- \(P = P_- \cup P_+\).
- Let \(p \in U' \cap P_-\).

Create in the search tree two children of \(<N, k>\).

- \(<N_1, k_1>\): \(N_1 = (P_1, C_1)\) with \(U'_1 = U' \setminus \{p\}\), 
  \(P'_1 = P' \cup \{p\}\), \(R'_1 = R'\), \(k_1 = k\)

- \(<N_2, k_2>\): \(N_2 = (P_2, C_2)\) with \(U'_2 = U' \setminus \{p\}\), 
  \(P'_2 = P'\), \(R'_2 = R' \cup \{p\}\), \(k_2 = k\).
How big is the search tree after applying the reduction rule as often as possible?
How big is the search tree after applying the reduction rule as often as possible?
An \textit{FPT}-Algorithm for $|C|-$

Annotated Coherence

- $2^{|P|-} \leq ?$
- $|P|- \leq 2|E|-$
- $2^{|P|-} \leq 2^2|E|-$
- Running time so far: $2^2|E|-|N|$
- If not solved: How does an instance look like after this branching process?
An *FPT*-Algorithm for \(|C|\)-Annotated Coherence

- In \(N\) are only vertices from \(P^{+}\) left! That means we are left with only positive undecided constraints.
- We can clean up the decided constraints, i.e. we remove them from the network.
- Afterwards we can also remove the isolated vertices.
Input: A network $N = (P, C)$. Here, $P$ is partitioned into $U^*$, $P^*$, and $R^*$, and $C = C^+$. A positive integer $k$.

Parameter: $|C^-|$

Question: Does there exist a partition of $P$ into $P'$ and $R$ such that $P^* \subseteq P'$, $R' \subseteq R$, and at least $k$ edges are satisfied by $A$ and $R$?
Let $p$ be a vertex where all neighbors are already selected. If $|N(v) \cap A^*| > |N(v) \cap R^*|$ then accept $p$, else reject $p$. 
If all nodes that are selected are in $A^*$, then $P = A$.

If all nodes that are selected are in $R^*$, then $P = R$.

Otherwise network is inconsistent.
Corollary: $|C^+|\text{-Coherence} \notin FPT$

(unless $P = NP$)
Parameterized Reduction