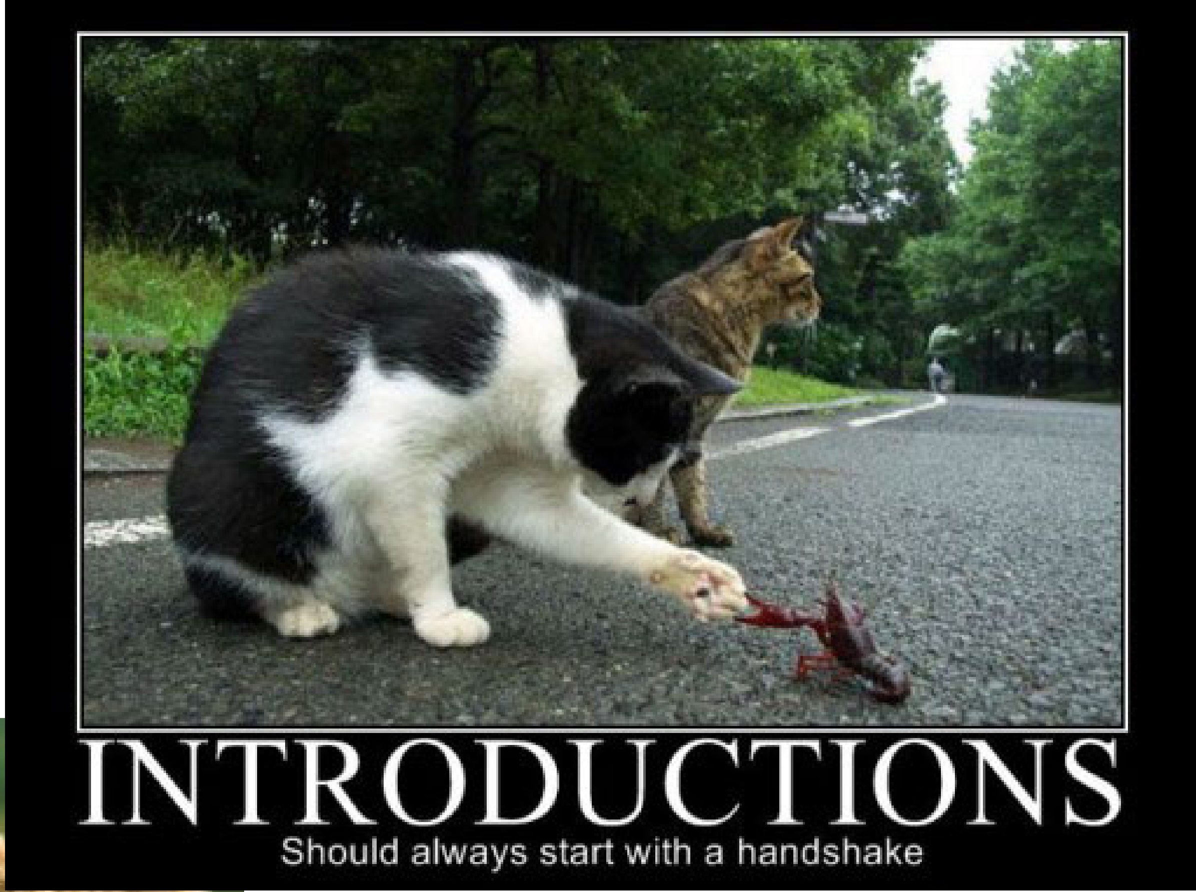
How (not) to do Introductions



by Thomas Bläsius

pictures stolen from the Internet

Warning

This presentation contains

- few facts,
- some bold claims,
- many unproven conjectures,
- maybe even some barefaced lies.

Feel free to disagree and discuss.

Warning

This presentation contains

- few facts,
- some bold claims,
- many unproven conjectures,
- maybe even some barefaced lies.

Feel free to disagree and discuss.

Claim

Writing a good introduction/paper is not really a writing skill.
It is mainly a reading skill.

1. Introduction

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

Do you like this Intro?
And if so, why not?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem STATION COVER that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for ot

Fact
The intro is typically less formal. Does NOT mean, you can write whatever you want.

the problem is not defining anything
you are defining the problem

Real-world data can often be modeled as a graph, e.g., social networks, biological ... ering problems on such networks are often ... vorks coming from public transit systems. ... can potentially help to let these systems run properly, which is importan for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap ... more efficient algorithms, also for ot...

Fact

The intro is typically less formal. Does NOT mean, you can write whatever you want.

Real-world data can often be modeled as a graph, e.g., social networks, biological ... ering problems on such networks are often ... vorks coming from public transit systems, ... can poten... run properly, which is importan... for a reliable, fast, and safe transport of passenge... Weihe [1] considered the problem STATION COVER that defines connections (e.g., trai... or buses) as paths in a graph of stations. His algorithm selects the minimum numbe... of vertices, such that every path contains at least one selected vertex. Despite the fac... that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

the problem is not defining anything you are defining the problem

is this really an implication?

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap ... more efficient algorithms, also for ot...

Fact
The intro is typically less formal. Does NOT mean, you can write whatever you want.

Real-world data can often be modeled as a graph, e.g., social networks, biological

the problem is not defining anything
you are defining the problem

...ering problems on such networks are often ...works coming from public transit systems ... can poten...

is this really an implication?

run properly, which is importan... for a reliable, fast, and safe transport of passenge... Weihe [1] considered the problem STATION COVER that defines connections (e.g., trai... or buses) as paths in a graph of stations. His algorithm selects the minimum numbe... of vertices, such that every path contains at least one selected vertex. Despite the fac... that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural pr...erties that make them easier.

not the problem is unrealistic,
the instances are

...applying a simple set of reduction rules and running ...maining instance. On real-world instances, apply- ... leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for ot...

Fact
The intro is typically less formal. Does NOT mean, you can write whatever you want.

maybe a bit over the top

R... be modeled as a graph, e.g., social networks, biological ... networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers.

Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap b... more efficient algorithms, also for ot...

Fact

The intro is typically less formal. Does NOT mean, you can write whatever you want.

maybe a bit over the top

… be modeled as a graph, e.g., social networks, biological … networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers.

Weihe [1] considered the problem Station Cover that defines connections (e.g., trains …

be specific: Station Cover is NP-hard

… His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for ot…

Fact

The intro is typically less formal. Does NOT mean, you can write whatever you want.

be modeled as a graph, e.g., social networks, biological

maybe a bit over the top

networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers.

Weihe [1] considered the problem Station Cover that defines connections (e.g., trains

be specific: Station Cover is NP-hard

His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instan

it is actuall just two rules

ural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for ot

Fact

The intro is typically less formal. Does NOT mean, you can write whatever you want.

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infra

is this relevant here?

g from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for ot

Fact
The intro is typically less formal. Does NOT mean, you can write whatever you want.

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infra[illegible] from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for ot[illegible]

is this relevant here?

Fact
The intro is typically less formal. Does NOT mean, you can write whatever you want.

Conjecture

The introduction is lacking structure/there is no golden thread.

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

the problem is hard (theory)

Conjecture

The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

the problem is hard (theory)

we consider setting x

Conjecture

The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

the problem is hard (theory)

we consider setting x

we consider an important problem

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

the problem is hard (theory)

we consider setting x

we consider an important problem

we consider problem x

Conjecture

The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem STATION COVER that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

the problem is hard (theory)

we consider setting x

we consider an important problem

we consider problem x

the problem is hard (theory)

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

the problem is hard (theory)

we consider setting x

we consider an important problem

we consider problem x

the problem is hard (theory)

the problem is easy (practice)

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

the problem is hard (theory)

we consider setting x

we consider an important problem

we consider problem x

the problem is hard (theory)

the problem is easy (practice)

realistic instances are easier

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem STATION COVER that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.
Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.
We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem
the problem is hard (theory)
we consider setting x
we consider an important problem
we consider problem x
the problem is hard (theory)
the problem is easy (practice)
realistic instances are easier
the algorithm is easy

Conjecture

The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

the problem is hard (theory)

we consider setting x

we consider an important problem

we consider problem x

the problem is hard (theory)

the problem is easy (practice)

realistic instances are easier

the algorithm is easy

the problem is easy (practice)

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.
Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.
We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

we consider an important problem

the problem is hard (theory)

we consider setting x

we consider an important problem

we consider problem x

the problem is hard (theory)

the problem is easy (practice)

realistic instances are easier

the algorithm is easy

the problem is easy (practice)

would be cool to close the gap

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem STATION COVER that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

- we consider an important problem
- the problem is hard (theory)
- we consider setting x
- we consider an important problem
- we consider problem x
- the problem is hard (theory)
- the problem is easy (practice)
- realistic instances are easier
- the algorithm is easy
- the problem is easy (practice)
- would be cool to close the gap
- realistic instances are easier

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier. Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

- we consider an important problem
- the problem is hard (theory)
- we consider setting x
- we consider an important problem
- we consider problem x
- the problem is hard (theory)
- the problem is easy (practice)
- realistic instances are easier
- the algorithm is easy
- the problem is easy (practice)
- would be cool to close the gap
- realistic instances are easier
- would be cool to close the gap

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem STATION COVER that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

- we consider an important problem
- the problem is hard (theory)
- we consider setting x
- we consider an important problem
- we consider problem x
- the problem is hard (theory)
- the problem is easy (practice)
- realistic instances are easier
- the algorithm is easy
- the problem is easy (practice)
- would be cool to close the gap
- realistic instances are easier
- would be cool to close the gap
- list of results

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

My suggestion

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem Station Cover that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

My suggestion

- we consider an important problem
- we consider problem x
- the problem is hard (theory)
- the problem is easy (practice)
- the algorithm is easy
- realistic instances are easier
- would be cool to close the gap
- list of results

Conjecture
The introduction is lacking structure/there is no golden thread.

What is each sentence actually saying?

Real-world data can often be modeled as a graph, e.g., social networks, biological networks, or infrastructure networks. Covering problems on such networks are often NP-hard. We consider infrastructure networks coming from public transit systems. Solving covering problems in such graphs can potentially help to let these systems run properly, which is important for a reliable, fast, and safe transport of passengers. Weihe [1] considered the problem STATION COVER that defines connections (e.g., trains or buses) as paths in a graph of stations. His algorithm selects the minimum number of vertices, such that every path contains at least one selected vertex. Despite the fact that covering problems are typically NP-hard, his algorithm is surprisingly fast. Thus, the algorithm uses the fact that the NP-hardness is usually based on an unrealistic variant of the problem, while real-world instances typically have certain structural properties that make them easier.

Weihe's algorithm works by first applying a simple set of reduction rules and running a brute-force algorithm on the remaining instance. On real-world instances, applying the reduction rules typically leads to a surprisingly small core. This raises the question, why these reduction rules are so effective. One approach to explain this, is to consider parameterized algorithms that exploit structural properties of the input. Such an algorithm can run in time $f(k)n^{O(1)}$, where k is a parameter that is assumed to be small for realistic instances. Understanding why the reduction rules are so effective is important to close the gap between theory and practice and can help to get more efficient algorithms, also for other covering problems.

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

My suggestion

- we consider an important problem
- we consider problem x
- the problem is hard (theory)
- the problem is easy (practice)
- the algorithm is easy
- realistic instances are easier
- would be cool to close the gap
- list of results

Claim
The introduction is actually lacking a motivation.

Claim
The introduction is actually lacking a motivation.

When one motivation is not enough

- we motivate the problem
- we motivate the research question

we consider an important problem

would be cool to close the gap

Claim
The introduction is actually lacking a motivation.

When one motivation is not enough

- we motivate the problem
- we motivate the research question
- we don't tell the reader, how our results answer it

we consider an important problem

would be cool to close the gap

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to the 2-core. Beyond that, we show that the problem remains NP-hard even for graphs of treewidth 3. We observe that real-world instances are heterogeneous and have high clustering. Thus, we run experiments on generated instances with varying heterogeneity and clustering. We observe that both properties help the reduction rules to be effective but the clustering appears to be the deciding factor.

Claim
The introduction is actually lacking a motivation.

When one motivation is not enough

- we motivate the problem — we consider an important problem
- we motivate the research question — would be cool to close the gap
- we don't tell the reader, how our results answer it

We show that the reduction rules reduce the graph at least to its 2-core. Moreover, for every graph, there exists an instance that is completely solved by the reduction rules. On the other hand, there is an instance for every graph where the reduction rules only lead to … even for graphs … ous and have hi… varying heterog… duction rules to …