Improving local search using parameterized complexity

Dániel Marx

Budapest University of Technology and Economics

dmarx@cs.bme.hu

Joint work with

Andrei Krokhin

Cork Constraint Computing Center

University College Cork, Ireland

December 10, 2008

Overview

- Local search algorithms
- Parameterized complexity approach to local search
- Applying this approach for the problem of finding minimum weight solutions for Boolean CSP's.
- Main result: classification theorem.

Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.

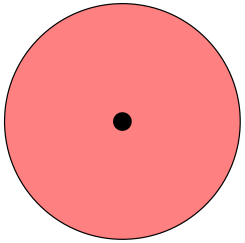
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.

-

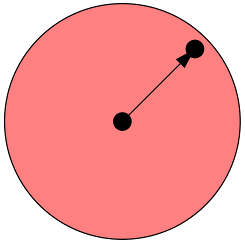
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



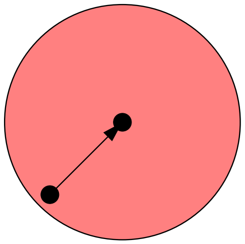
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



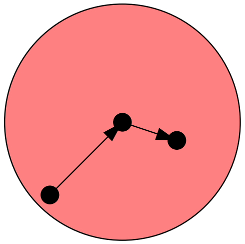
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



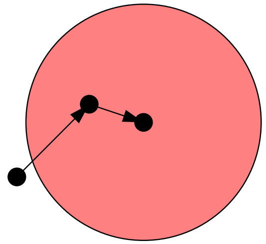
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



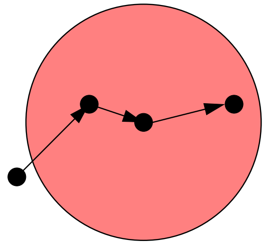
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



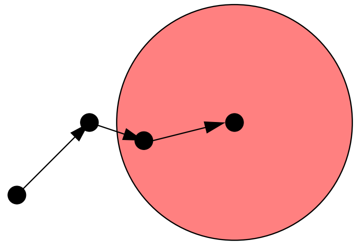
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



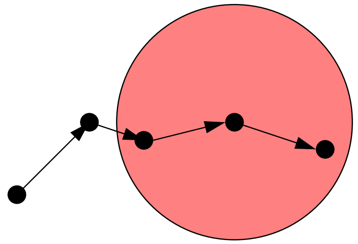
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



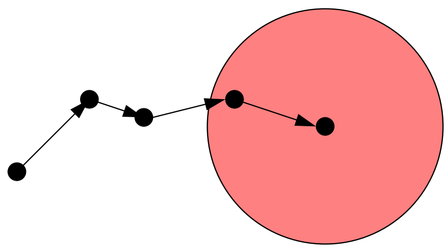
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



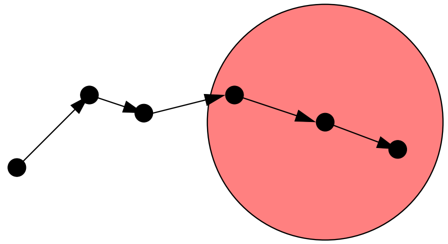
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



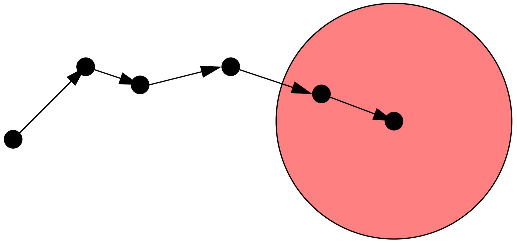
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



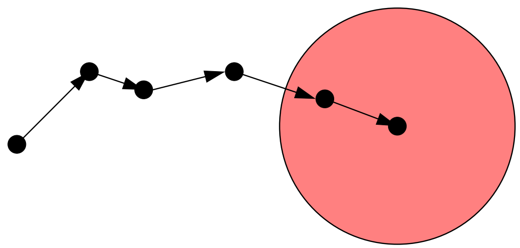
Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



Local search

Local search: walk in the solution space by iteratively replacing the current solution with a better solution in the local neighborhood.



Problem: local search can stop at a local optimum (no better solution in the local neighborhood).

More sophisticated variants: simulated annealing, tabu search, etc.

Local neighborhood

The local neighborhood is defined in a problem-specific way:

- For TSP, the neighbors are obtained by swapping 2 cities or replacing 2 edges.
- For a problem with 0-1 variables, the neighbors are obtained by flipping a single variable.
- For subgraph problems, the neighbors are obtained by adding/removing one edge.

Local neighborhood

The local neighborhood is defined in a problem-specific way:

- For TSP, the neighbors are obtained by swapping 2 cities or replacing 2 edges.
- For a problem with 0-1 variables, the neighbors are obtained by flipping a single variable.
- For subgraph problems, the neighbors are obtained by adding/removing one edge.

More generally: reordering k cities, flipping k variables, etc.

Larger neighborhood (larger k):

- algorithm is less likely to get stuck in a local optimum,
- it is more difficult to check if there is a better solution in the neighborhood.

Searching the neighborhood

Is there an efficient way of finding a better solution in the k-neighborhood?

We study the complexity of the following problem:

Input:	instance I, solution x, integer k
Decide:	Is there a solution x' with dist$(x, x') \leq k$ that is "better" than x?

Searching the neighborhood

Is there an efficient way of finding a better solution in the k-neighborhood?

We study the complexity of the following problem:

Input:	instance I, solution x, integer k
Decide:	Is there a solution x' with dist$(x, x') \leq k$ that is "better" than x?

Remark 1: If the optimization problem is hard, then it is unlikely that this local search problem is polynomial-time solvable: otherwise we would be able to test if a solution is optimal.

Searching the neighborhood

Is there an efficient way of finding a better solution in the k-neighborhood?
We study the complexity of the following problem:

Input:	instance I, solution x, integer k
Decide:	Is there a solution x' with $\text{dist}(x, x') \leq k$ that is "better" than x?

Remark 1: If the optimization problem is hard, then it is unlikely that this local search problem is polynomial-time solvable: otherwise we would be able to test if a solution is optimal.

Remark 2: Size of the k-neighborhood is usually $n^{O(k)}$ $\Rightarrow$ local search is polynomial-time solvable for every fixed k, but it is not practical for larger k.

Searching the neighborhood

Is there an efficient way of finding a better solution in the k-neighborhood?
We study the complexity of the following problem:

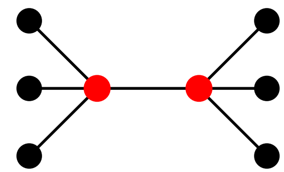
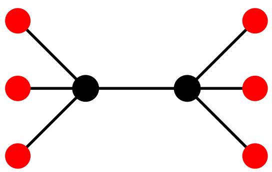
Input:	instance I, solution x, integer k
Decide:	Is there a solution x' with $\text{dist}(x, x') \le k$ that is "better" than x?

Remark 1: If the optimization problem is hard, then it is unlikely that this local search problem is polynomial-time solvable: otherwise we would be able to test if a solution is optimal.

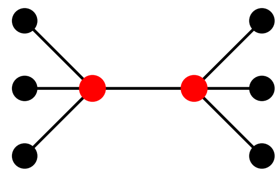
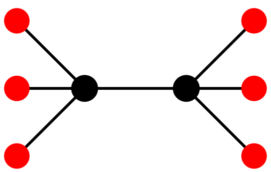
Remark 2: Size of the k-neighborhood is usually $n^{O(k)}$ $\Rightarrow$ local search is polynomial-time solvable for every fixed k, but it is not practical for larger k.

Classical complexity theory does not tell us anything useful about the complexity of local search!

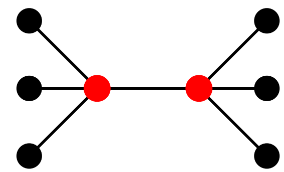
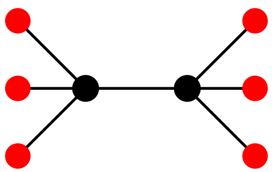
Parameterized complexity

Problem:	MINIMUM VERTEX COVER	MAXIMUM INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices? 	Is it possible to find k independent vertices? 
Complexity:	NP-complete	NP-complete

Parameterized complexity

Problem:	MINIMUM VERTEX COVER	MAXIMUM INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?
		
Complexity:	NP-complete	NP-complete
Complete enumeration:	$O(n^k)$ possibilities	$O(n^k)$ possibilities

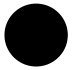
Parameterized complexity

Problem:	MINIMUM VERTEX COVER	MAXIMUM INDEPENDENT SET
Input:	Graph G, integer k	Graph G, integer k
Question:	Is it possible to cover the edges with k vertices?	Is it possible to find k independent vertices?
		
Complexity:	NP-complete	NP-complete
Complete enumeration:	$O(n^k)$ possibilities	$O(n^k)$ possibilities
	$O(2^k n^2)$ algorithm exists	No $n^{o(k)}$ algorithm known
		

Bounded search tree method

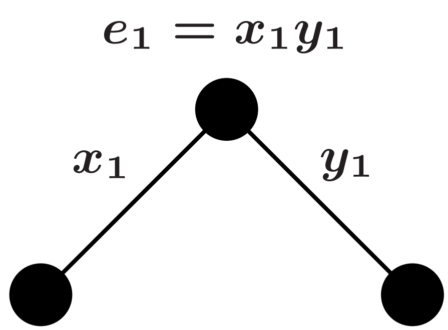
Algorithm for MINIMUM VERTEX COVER:

$$e_1 = x_1 y_1$$



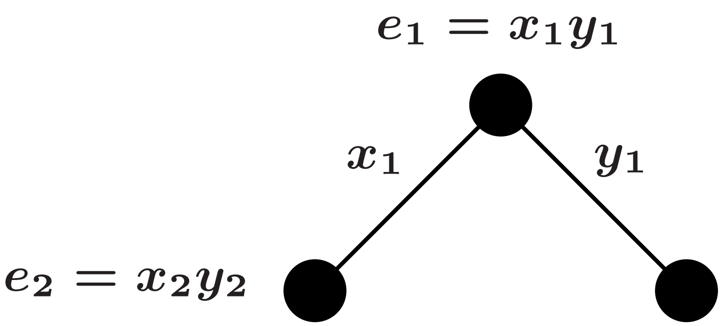
Bounded search tree method

Algorithm for MINIMUM VERTEX COVER:



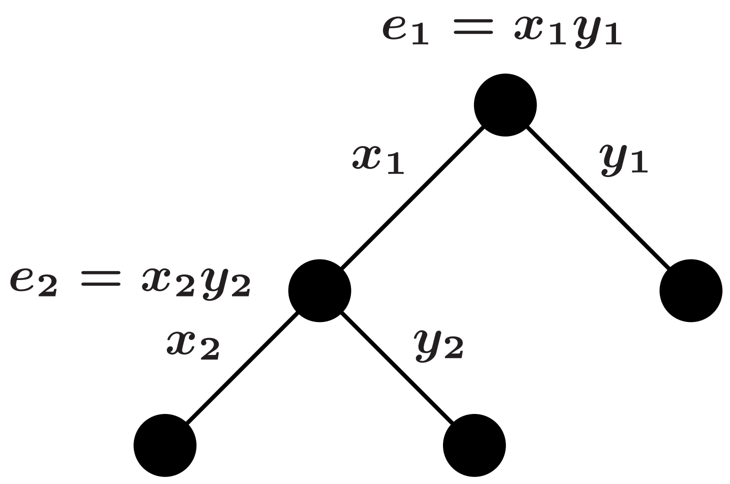
Bounded search tree method

Algorithm for MINIMUM VERTEX COVER:



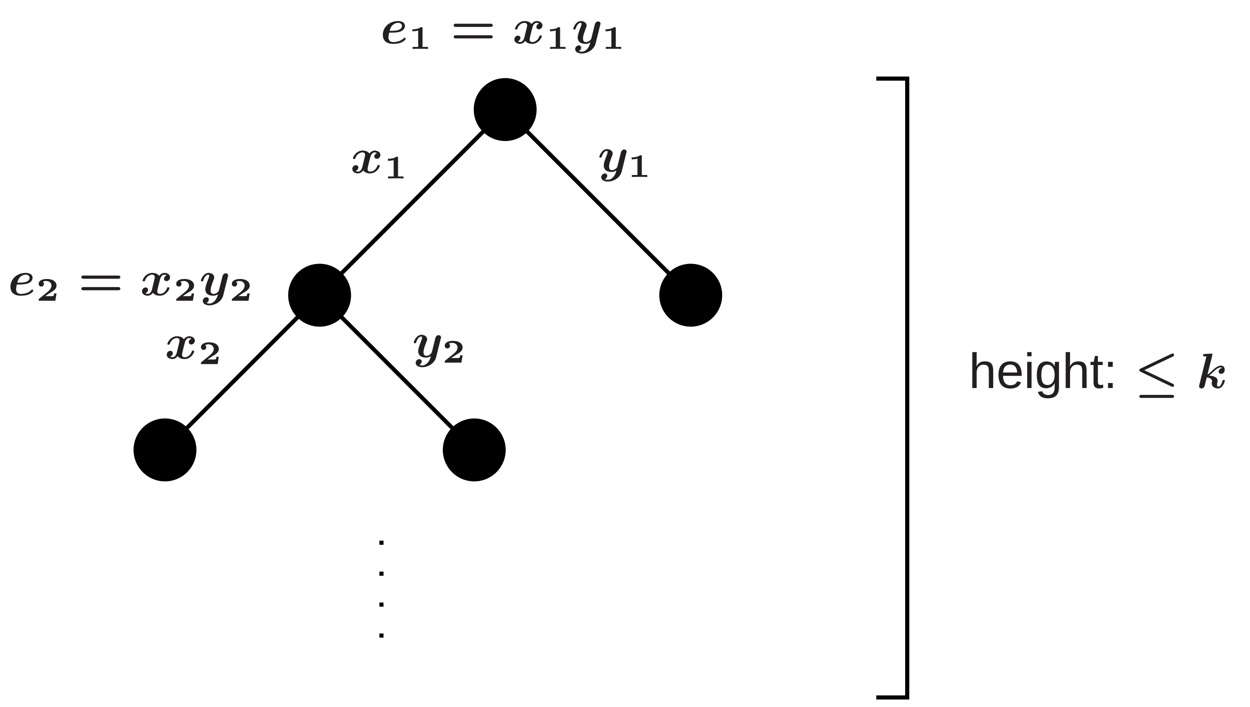
Bounded search tree method

Algorithm for MINIMUM VERTEX COVER:



Bounded search tree method

Algorithm for MINIMUM VERTEX COVER:



Height of the search tree is $\leq k \Rightarrow$ number of nodes is $O(2^k) \Rightarrow$ complete search requires $2^k \cdot$ poly steps.

Fixed-parameter tractability

Definition: a **parameterized problem** is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

We have seen that MINIMUM VERTEX COVER is in FPT. Best known algorithm: $O(1.2832^k k + k|V|)$ [Niedermeier, Rossmanith, 2003]

Main goal of parameterized complexity: to find FPT problems.

Fixed-parameter tractability

Definition: a **parameterized problem** is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

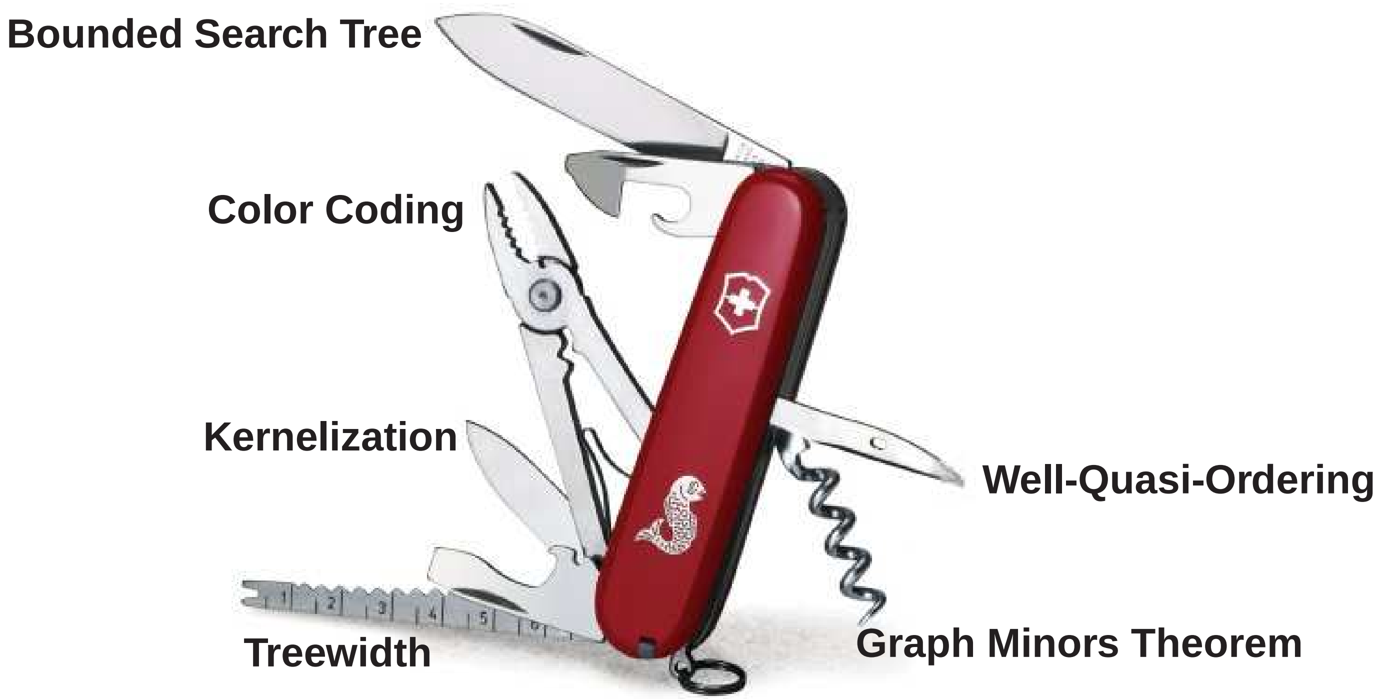
We have seen that MINIMUM VERTEX COVER is in FPT. Best known algorithm: $O(1.2832^k k + k|V|)$ [Niedermeier, Rossmanith, 2003]

Main goal of parameterized complexity: to find FPT problems.
Examples of NP-hard problems that are FPT:

- Finding a vertex cover of size k.
- Finding a path of length k.
- Finding k disjoint triangles.
- Drawing the graph in the plane with k edge crossing.
- Finding disjoint paths that connect k pairs of points.
- ...

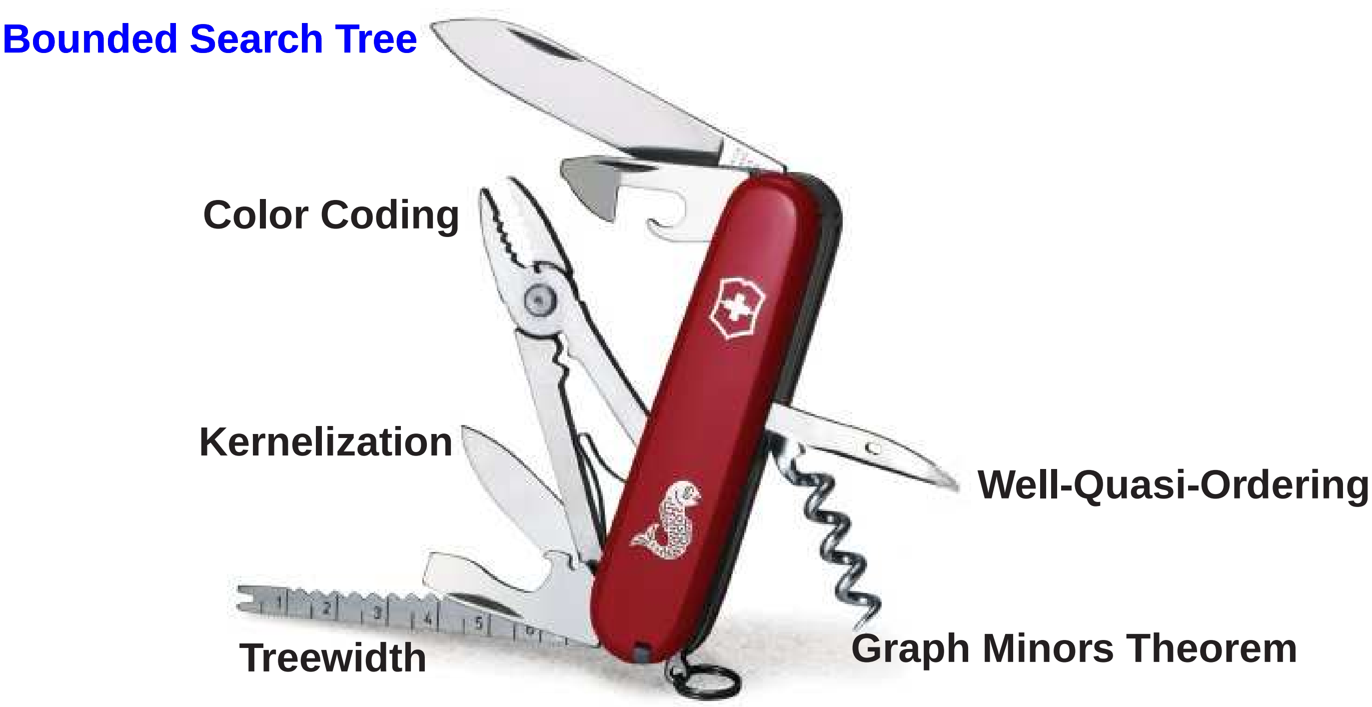
Fixed-parameter tractability (cont.)

- Practical importance: efficient algorithms for small values of k.
- Powerful toolbox for designing FPT algorithms:



Fixed-parameter tractability (cont.)

- Practical importance: efficient algorithms for small values of k.
- Powerful toolbox for designing FPT algorithms:



Parameterized intractability

We expect that MAXIMUM INDEPENDENT SET is not fixed-parameter tractable, no $n^{o(k)}$ algorithm is known.

W[1]-complete $\approx$ "as hard as MAXIMUM INDEPENDENT SET"

Parameterized intractability

We expect that MAXIMUM INDEPENDENT SET is not fixed-parameter tractable, no $n^{o(k)}$ algorithm is known.

W[1]-complete $\approx$ "as hard as MAXIMUM INDEPENDENT SET"

Parameterized reductions: L_1 is reducible to L_2, if there is a function f that transforms (x, k) to (x', k') such that

- $(x, k) \in L_1$ if and only if $(x', k') \in L_2$,
- f can be computed in $f(k)|x|^c$ time,
- **k' depends only on k**

If L_1 is reducible to L_2, and L_2 is in FPT, then L_1 is in FPT as well.

Most NP-completeness proofs are not good for parameterized reductions.

Parameterized Complexity: Summary

Two key concepts:

- A parameterized problem is **fixed-parameter tractable** if it has an $f(k)n^c$ time algorithm.
- To show that a problem L is hard, we have to give a **parameterized reduction** from a known **W[1]-complete** problem to L.

Parameterized Complexity: Summary

Two key concepts:

- A parameterized problem is **fixed-parameter tractable** if it has an $f(k)n^c$ time algorithm.
- To show that a problem L is hard, we have to give a **parameterized reduction** from a known **W[1]-complete** problem to L.

The question that we want to investigate:

$\boxed{\text{Is } k\text{-local-search fixed-parameter tractable for a particular problem?}}$

If yes, then local search algorithms can consider larger neighborhoods, improving their efficiency.

Important: k is the number of allowed changes and **not** the size of the solution. Relevant even if solution size is large.

Results on parameterized local search

- **Task:** find a spanning tree maximizing the number of vertices having full degree.

  Local search is FPT: given a solution, it can be checked in time $O(n^2 + nf(k))$ if it is possible to obtain a better solution by replacing at most k edges [Khuller, Bhatia, and Pless 2003].

- **Task:** TSP with distances satisfying the triangle inequality.

  Local search is hard: it is W[1]-hard to check if it is possible to obtain a shorter tour by replacing at most k arcs [M. 2008].

Results on parameterized local search (cont.)

- **Task:** find a minimum dominating set/minimum r-center/minimum vertex cover in a planar graph.

  Local search is FPT. [Fellows et al., 2008].

Results on parameterized local search (cont.)

- **Task:** find a minimum dominating set/minimum r-center/minimum vertex cover in a planar graph.

  Local search is FPT. [Fellows et al., 2008].

- **Task:** find a maximum stable assignment in the “Hospitals/Residents with Couples” problem (a variant of Stable Marriage).

  - Local search is W[1]-hard:
    There is no $f(k) \cdot n^{O(1)}$ algorithm for deciding whether an assignment can be improved by at most k changes.
  - Local search is FPT if the number ℓ of couples is also a parameter:
    There is an $f(k, \ell) \cdot n^{O(1)}$ for deciding whether an assignment can be improved by at most k changes. [M. and Schlotter 2008].

Boolean CSP

Topic of this talk: investigating the parameterized complexity of local search for the problem of finding a minimum weight solution for a Boolean constraint satisfaction problem (CSP).

Boolean CSP: generalization of SAT. Input is a conjunction of constraints over a set of Boolean variables.

$$R_1(x_1, x_4, x_5) \wedge R_2(x_2, x_1) \wedge R_1(x_3, x_3, x_3) \wedge R_3(x_5, x_1, x_4, x_1)$$

Constraints can be arbitrary Boolean relations.

Problem is too general!

Boolean CSP

If Γ is a set of Boolean relations, then a **Γ-formula** is a conjunction of relations in Γ:

$$R_1(x_1, x_4, x_5) \wedge R_2(x_2, x_1) \wedge R_1(x_3, x_3, x_3) \wedge R_3(x_5, x_1, x_4, x_1)$$

Γ-SAT

- Given: an Γ-formula φ
- Find: a variable assignment satisfying φ

Boolean CSP

If Γ is a set of Boolean relations, then a **Γ-formula** is a conjunction of relations in Γ:

$$R_1(x_1, x_4, x_5) \wedge R_2(x_2, x_1) \wedge R_1(x_3, x_3, x_3) \wedge R_3(x_5, x_1, x_4, x_1)$$

Γ-SAT

- Given: an Γ-formula φ
- Find: a variable assignment satisfying φ

$\Gamma = \{a \neq b\} \Rightarrow$ Γ-SAT = 2-coloring of a graph
$\Gamma = \{a \vee b,\ a \vee \bar{b},\ \bar{a} \vee \bar{b}\} \Rightarrow$ Γ-SAT = 2SAT
$\Gamma = \{a \vee b \vee c, a \vee b \vee \bar{c}, a \vee \bar{b} \vee \bar{c}, \bar{a} \vee \bar{b} \vee \bar{c}\} \Rightarrow$ Γ-SAT = 3SAT

Boolean CSP

If Γ is a set of Boolean relations, then a **Γ-formula** is a conjunction of relations in Γ:

$$R_1(x_1, x_4, x_5) \wedge R_2(x_2, x_1) \wedge R_1(x_3, x_3, x_3) \wedge R_3(x_5, x_1, x_4, x_1)$$

Γ-SAT

- Given: an Γ-formula φ
- Find: a variable assignment satisfying φ

$\Gamma = \{a \neq b\} \Rightarrow \Gamma$-SAT = 2-coloring of a graph
$\Gamma = \{a \vee b,\ a \vee \bar{b},\ \bar{a} \vee \bar{b}\} \Rightarrow \Gamma$-SAT = 2SAT
$\Gamma = \{a \vee b \vee c, a \vee b \vee \bar{c}, a \vee \bar{b} \vee \bar{c}, \bar{a} \vee \bar{b} \vee \bar{c}\} \Rightarrow \Gamma$-SAT = 3SAT

Question: Γ-SAT is polynomial time solvable for which Γ?
It is NP-complete for which Γ?

Schaefer's Dichotomy Theorem (1978)

For every finite Γ, the Γ-SAT problem is polynomial time solvable if one of the following holds, and NP-complete otherwise:

- Every relation is satisfied by the all 0 assignment
- Every relation is satisfied by the all 1 assignment
- Every relation can be expressed by a 2SAT formula
- Every relation can be expressed by a Horn formula
- Every relation can be expressed by an anti-Horn formula
- Every relation is an affine subspace over $GF(2)$

Other dichotomy results

- Approximability of MAX-SAT, MIN-UNSAT [Khanna et al., 2001]
- Approximability of MAX-ONES, MIN-ONES [Khanna et al., 2001]
- Generalization to 3 valued variables [Bulatov, 2002]
- Inverse satisfiability [Kavvadias and Sideri, 1999]
- Parameterized complexity of weight k solutions [M., 2005]
- Counting solutions [Bulatov, 2008]
- etc.

Minimizing weight

Γ-MIN-ONES: find a solution of a Γ-SAT formula that minimizes the weight (= the number of 1's).

Theorem: [Khanna et al., 2001] For every finite Γ, the Γ-MIN-ONES problem is polynomial time solvable if one of the following holds, and NP-complete otherwise:

- Every relation is satisfied by the all 0 assignment
- Every relation can be expressed by a Horn formula
- Every relation is width-2 affine (= can be expressed by constants, $=$, $\neq$).

Our goal: characterize those sets Γ where local search for Γ-MIN-ONES is fixed-parameter tractable.

Losing weight

$\mathbf{\Gamma}$-LOSE-WEIGHT	
Input:	A $\mathbf{\Gamma}$-formula φ, a solution x for φ, and an integer k.
Decide:	Is there a solution x' of φ with dist$(x, x') \leq k$ and weight$(x') <$ weight(x)?

dist(x, x'): Hamming distance of x and x'.

weight(x): number of 1's in x.

Losing weight

Γ-LOSE-WEIGHT

Input:	A Γ-formula φ, a solution x for φ, and an integer k.
Decide:	Is there a solution x' of φ with $\mathrm{dist}(x, x') \leq k$ and $\mathrm{weight}(x') < \mathrm{weight}(x)$?

$\mathrm{dist}(x, x')$: Hamming distance of x and x'.
$\mathrm{weight}(x)$: number of 1's in x.

Main result:

Theorem: For every finite set Γ, Γ-LOSE-WEIGHT is either fixed-parameter tractable or W[1]-hard.

+ a simple characterization of the FPT cases.

Horn constraints

Definition: A relation is **Horn** (or weakly negative) if it can be expressed as the conjunction of clauses with at most one positive literal in each clause.

$$(x_1 \vee \bar{x}_2) \wedge (x_3) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_2)$$

A relation is Horn if and only if it is closed under componentwise AND.

Flip sets

Definition: Let R be an r-ary relation and $(a_1, \ldots, a_r) \in R$. A set $S \subseteq \{1, \ldots, r\}$ is a **flip set** of $(a_1, \ldots, a_r)$ (with respect to R) if flipping the coordinates corresponding to S gives another tuple in R.

Example:

$$R(x_1, x_2, x_3, x_4)$$
$$(0, 0, 1, 0)$$
$$(1, 0, 1, 0)$$
$$(0, 1, 1, 1)$$
$$(1, 0, 0, 0)$$
$$(0, 1, 1, 0)$$
$$(1, 0, 1, 1)$$

Flip sets

Definition: Let R be an r-ary relation and $(a_1, \ldots, a_r) \in R$. A set $S \subseteq \{1, \ldots, r\}$ is a **flip set** of $(a_1, \ldots, a_r)$ (with respect to R) if flipping the coordinates corresponding to S gives another tuple in R.

Example:

$R(x_1, x_2, x_3, x_4)$	Flip sets of $(1, 0, 1, 0)$
$(0, 0, 1, 0)$	$\{1\}$
$(1, 0, 1, 0)$	
$(0, 1, 1, 1)$	$\{1, 2, 4\}$
$(1, 0, 0, 0)$	$\{3\}$
$(0, 1, 1, 0)$	$\{1, 2\}$
$(1, 0, 1, 1)$	$\{4\}$

Flip sets

Definition: Let R be an r-ary relation and $(a_1, \ldots, a_r) \in R$. A set $S \subseteq \{1, \ldots, r\}$ is a **flip set** of $(a_1, \ldots, a_r)$ (with respect to R) if flipping the coordinates corresponding to S gives another tuple in R.

Example:

$R(x_1, x_2, x_3, x_4)$	Flip sets of $(1,0,1,0)$	Flip sets of $(0,1,1,1)$
$(0,0,1,0)$	$\{1\}$	$\{2,3\}$
$(1,0,1,0)$		$\{1,2,4\}$
$(0,1,1,1)$	$\{1,2,4\}$	
$(1,0,0,0)$	$\{3\}$	$\{1,2,3,4\}$
$(0,1,1,0)$	$\{1,2\}$	$\{4\}$
$(1,0,1,1)$	$\{4\}$	$\{1,2\}$

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example: Flip sets of

$R(x_1, x_2, x_3, x_4)$	$(1, 0, 1, 0)$
$(0, 0, 1, 0)$	$\{1\}$
$(1, 0, 1, 0)$	
$(0, 1, 1, 1)$	$\{1, 2, 4\}$
$(1, 0, 0, 0)$	$\{3\}$
$(0, 1, 1, 0)$	$\{1, 2\}$
$(1, 0, 1, 1)$	$\{4\}$

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

$R(x_1, x_2, x_3, x_4)$	Flip sets of $(1, 0, 1, 0)$	
$(0, 0, 1, 0)$	$\{1\}$	
$(1, 0, 1, 0)$		R is not
$(0, 1, 1, 1)$	$\{1, 2, 4\}$	flip separable!
$(1, 0, 0, 0)$	$\{3\}$	
$(0, 1, 1, 0)$	$\{1, 2\}$	
$(1, 0, 1, 1)$	$\{4\}$	

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

EVEN(x_1, x_2, x_3, x_4)

$(0,0,0,0)$

$(1,1,0,0)$

$(1,0,1,0)$

$(1,0,0,1)$

$(0,1,1,0)$

$(0,1,0,1)$

$(0,0,1,1)$

$(1,1,1,1)$

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

EVEN(x_1, x_2, x_3, x_4)	Flip sets of $(1,1,0,0)$
$(0,0,0,0)$	$\{1,2\}$
$(1,1,0,0)$	
$(1,0,1,0)$	$\{2,3\}$
$(1,0,0,1)$	$\{2,4\}$
$(0,1,1,0)$	$\{1,3\}$
$(0,1,0,1)$	$\{1,4\}$
$(0,0,1,1)$	$\{1,2,3,4\}$
$(1,1,1,1)$	$\{3,4\}$

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

EVEN(x_1, x_2, x_3, x_4)	Flip sets of $(1,1,0,0)$	Flip sets of $(1,1,1,1)$
$(0,0,0,0)$	$\{1,2\}$	$\{1,2,3,4\}$
$(1,1,0,0)$		$\{3,4\}$
$(1,0,1,0)$	$\{2,3\}$	$\{2,4\}$
$(1,0,0,1)$	$\{2,4\}$	$\{2,3\}$
$(0,1,1,0)$	$\{1,3\}$	$\{1,4\}$
$(0,1,0,1)$	$\{1,4\}$	$\{1,3\}$
$(0,0,1,1)$	$\{1,2,3,4\}$	$\{1,2\}$
$(1,1,1,1)$	$\{3,4\}$	

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

	Flip sets of	Flip sets of
EVEN(x_1, x_2, x_3, x_4)	$(1,1,0,0)$	$(1,1,1,1)$
$(0,0,0,0)$	$\{1,2\}$	$\{1,2,3,4\}$
$(1,1,0,0)$		$\{3,4\}$
$(1,0,1,0)$	$\{2,3\}$	$\{2,4\}$
$(1,0,0,1)$	$\{2,4\}$	$\{2,3\}$
$(0,1,1,0)$	$\{1,3\}$	$\{1,4\}$
$(0,1,0,1)$	$\{1,4\}$	$\{1,3\}$
$(0,0,1,1)$	$\{1,2,3,4\}$	$\{1,2\}$
$(1,1,1,1)$	$\{3,4\}$	

EVEN is flip separable!

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

1-IN-4(x_1, x_2, x_3, x_4)

$$(1, 0, 0, 0)$$
$$(0, 1, 0, 0)$$
$$(0, 0, 1, 0)$$
$$(0, 0, 0, 1)$$

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

	Flip sets of
1-IN-4(x_1, x_2, x_3, x_4)	$(1, 0, 0, 0)$
$(1, 0, 0, 0)$	
$(0, 1, 0, 0)$	$\{1, 2\}$
$(0, 0, 1, 0)$	$\{1, 3\}$
$(0, 0, 0, 1)$	$\{1, 4\}$

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

1-IN-4(x_1, x_2, x_3, x_4)	Flip sets of $(1,0,0,0)$	Flip sets of $(0,1,0,0)$
$(1,0,0,0)$		$\{1,2\}$
$(0,1,0,0)$	$\{1,2\}$	
$(0,0,1,0)$	$\{1,3\}$	$\{2,3\}$
$(0,0,0,1)$	$\{1,4\}$	$\{2,4\}$

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

1-IN-4(x_1, x_2, x_3, x_4)	Flip sets of $(1,0,0,0)$	Flip sets of $(0,1,0,0)$	
$(1,0,0,0)$		$\{1,2\}$	1-IN-4 is
$(0,1,0,0)$	$\{1,2\}$		flip separable!
$(0,0,1,0)$	$\{1,3\}$	$\{2,3\}$	
$(0,0,0,1)$	$\{1,4\}$	$\{2,4\}$	

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example:

$x_1 \vee x_2$

$(1, 0)$

$(0, 1)$

$(1, 1)$

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example: Flip sets of

$x_1 \vee x_2$	$(1,0)$
$(1,0)$	
$(0,1)$	$\{1,2\}$
$(1,1)$	$\{2\}$

Flip separable

Definition: An r-ary relation R is **flip separable** if whenever $S_1 \subset S_2 \subseteq \{1, \ldots, r\}$ are flip sets of a tuple $(x_1, \ldots, x_r)$, then $S_2 \setminus S_1$ is also a flip set.

Example: Flip sets of

$x_1 \vee x_2$	$(1,0)$
$(1,0)$	
$(0,1)$	$\{1,2\}$
$(1,1)$	$\{2\}$

$x_1 \vee x_2$ is not

flip separable!

Main result

Theorem: For every finite set Γ, Γ-LOSE-WEIGHT is fixed-parameter tractable if one of the following holds, and W[1]-hard otherwise:

- Every relation can be expressed by a Horn formula.
- Every relation is flip separable.

Some FPT cases:

- EVEN and ODD constraints.
- affine constraints.
- p-IN-q constraints.

Some hard cases:

- $x_1 \vee x_2$ (= MINIMUM VERTEX COVER)
- 3SAT

Algorithm

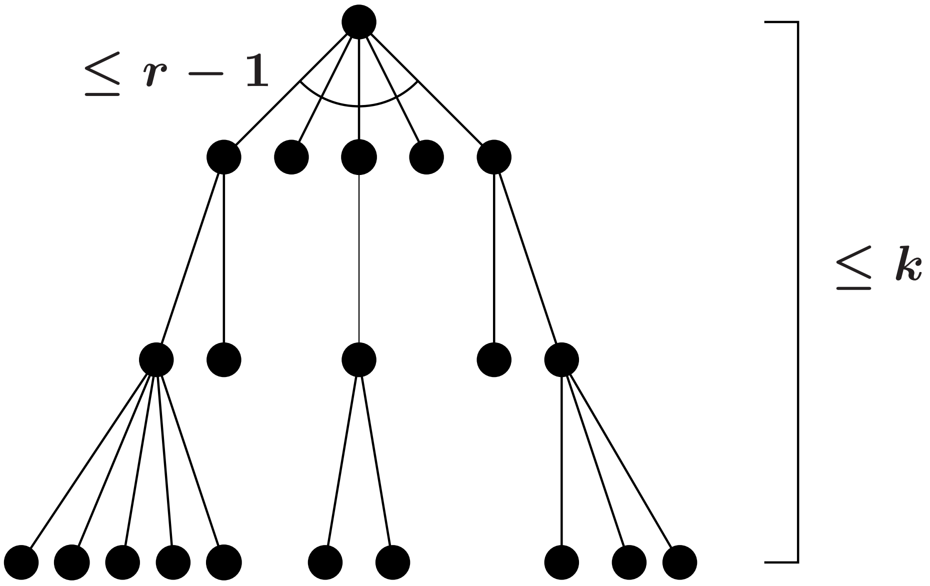
Task: given a formula with flip separable constraints and a satisfying assignment, decrease the weight by flipping at most k variables.

Bounded search tree algorithm:

- Flip a variable with value 1 to 0 (at most n possible choices).
- If a clause is not satisfied, flip one of its variables that was not yet flipped (at most $r - 1$ possible choices if maximum arity is r).
- Repeat until
  - more than k variables are flipped $\Rightarrow$ terminate this branch.
  - every clause is satisfied $\Rightarrow$ check if the satisfying assignment has strictly smaller weight than the original assignment.

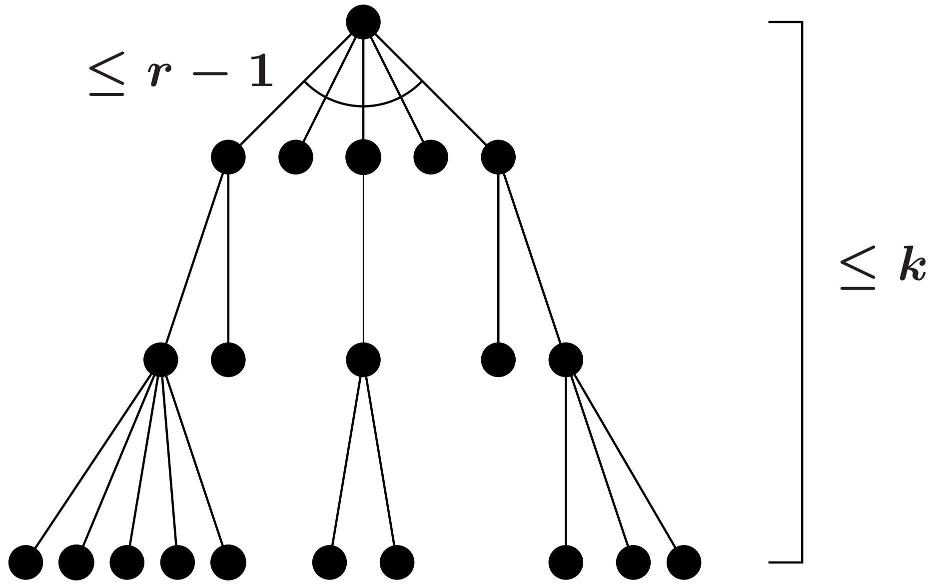
Algorithm

Running time: After the initial flip, the search tree has size at most $(r-1)^k$:



Algorithm

Running time: After the initial flip, the search tree has size at most $(r-1)^k$:



Running time is $f(k,r)\cdot n^c \Rightarrow f'(k)\cdot n^c$ for a fixed Γ.

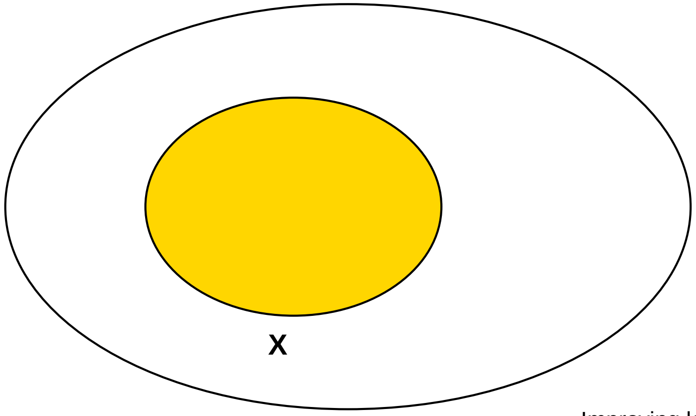
Algorithm

Correctness: is it true that we always find a solution if it exits?

Algorithm

Correctness: is it true that we always find a solution if it exits?

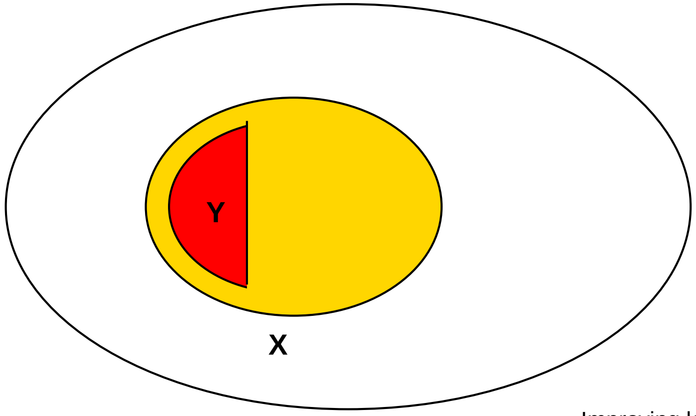
- Let X be a solution that decreases the weight most ($|X| \leq k$, flipping X gives a satisfying assignment).



Algorithm

Correctness: is it true that we always find a solution if it exits?

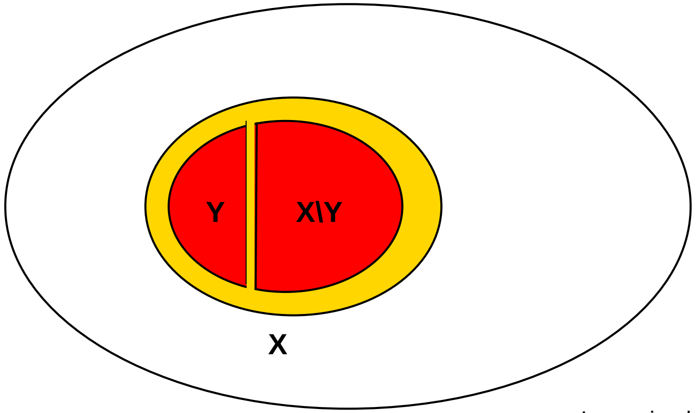
- Let X be a solution that decreases the weight most ($|X| \leq k$, flipping X gives a satisfying assignment).
- There is a branch of the algorithm that flips only a subset $Y \subseteq X$.



Algorithm

Correctness: is it true that we always find a solution if it exits?

- Let X be a solution that decreases the weight most ($|X| \leq k$, flipping X gives a satisfying assignment).
- There is a branch of the algorithm that flips only a subset $Y \subseteq X$.
- Flipping $X \setminus Y$ is also a solution (constraints are flip separable).
- If flipping Y does not decrease the weight, then flipping $X \setminus Y$ decreases the weight more than Y.



Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Step 1: Direct proof for $x \vee y$.

$\Rightarrow$ Given a vertex cover S and an integer k, it is W[1]-hard to decide if it is possible to decrease the vertex cover by adding/removing at most k vertices.

$\Rightarrow$ Given an independent set S and an integer k, it is W[1]-hard to decide if it is possible to increase the independent set cover by adding/removing at most k vertices.

Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Step 1: Direct proof for $x \vee y$.

$\Rightarrow$ Given a vertex cover S and an integer k, it is W[1]-hard to decide if it is possible to decrease the vertex cover by adding/removing at most k vertices.

$\Rightarrow$ Given an independent set S and an integer k, it is W[1]-hard to decide if it is possible to increase the independent set cover by adding/removing at most k vertices.

Note: These results hold even for bipartite graphs.

Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Step 2: Suppose that there is a relation $R \in \Gamma$ that is not Horn, i.e., it is not closed under componentwise AND.

Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Step 2: Suppose that there is a relation $R \in \Gamma$ that is not Horn, i.e., it is not closed under componentwise AND.

$(1, 0, 0, 1) \in R$
$(0, 1, 0, 1) \in R$
$(0, 0, 0, 1) \notin R$

Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Step 2: Suppose that there is a relation $R \in \Gamma$ that is not Horn, i.e., it is not closed under componentwise AND.

$(1, 0, 0, 1) \in R$
$(0, 1, 0, 1) \in R$
$(0, 0, 0, 1) \notin R$

either

$(1, 1, 0, 1) \in R$
$\Rightarrow R(x, y, 0, 1) \equiv x \vee y$, we can "almost express" relation $x \vee y$ (DONE).

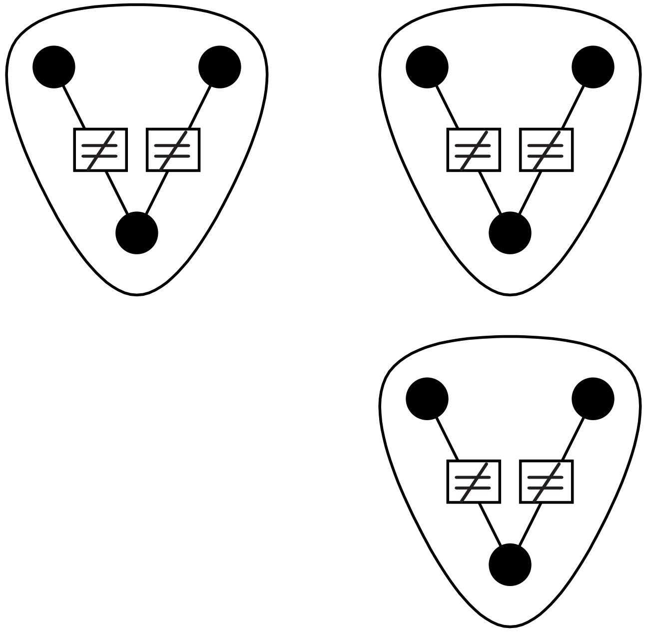
$(1, 1, 0, 1) \notin R$
$\Rightarrow R(x, y, 0, 1) \equiv x \neq y$, we can "almost express" relation $\neq$.

Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Step 3: Suppose that there is a relation $R \in \Gamma$ that is not flip separable and we can use $\neq$.

- Reduction from $x \vee y$.
- Replace each variable with 3 variables
- Two states for each triple.
- Changing a triple changes the weight by 1.

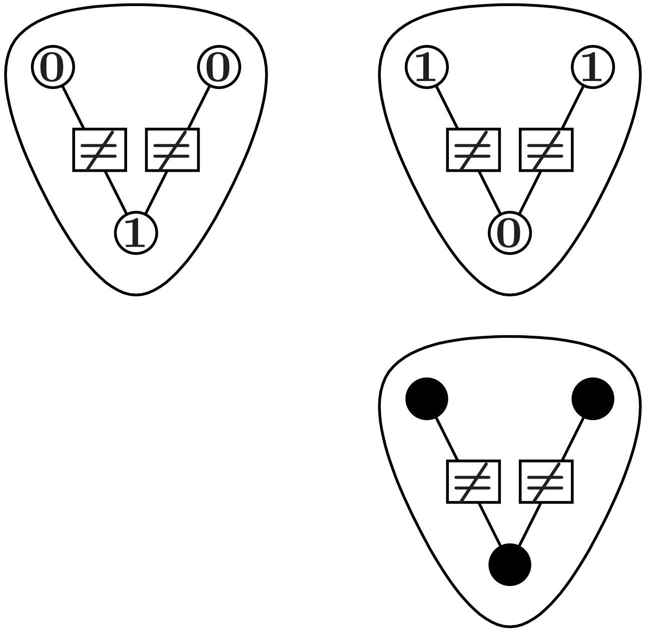


Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Step 3: Suppose that there is a relation $R \in \Gamma$ that is not flip separable and we can use $\neq$.

- Reduction from $x \vee y$.
- Replace each variable with 3 variables
- Two states for each triple.
- Changing a triple changes the weight by 1.

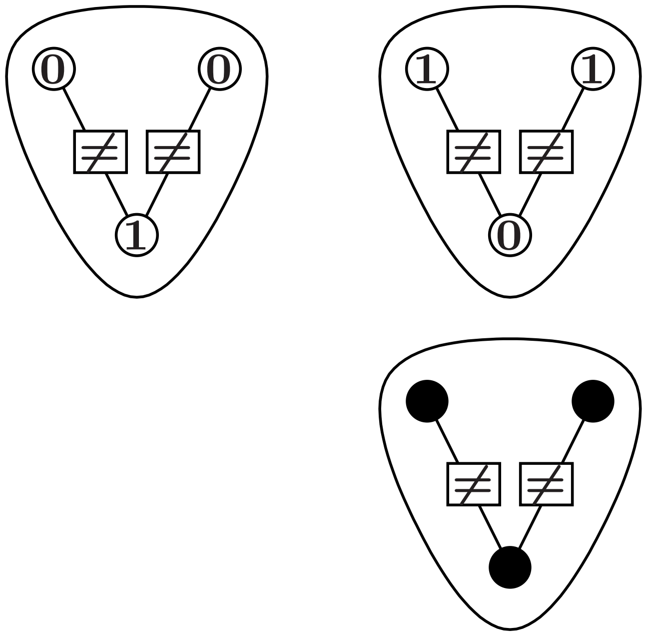


Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Suppose there is a counterexample to the fact that $R \in \Gamma$ is flip separable:

$(0,1,0,1) \in R$
$(1,0,0,1) \in R$
$(1,0,1,0) \in R$
$(0,1,1,0) \notin R$

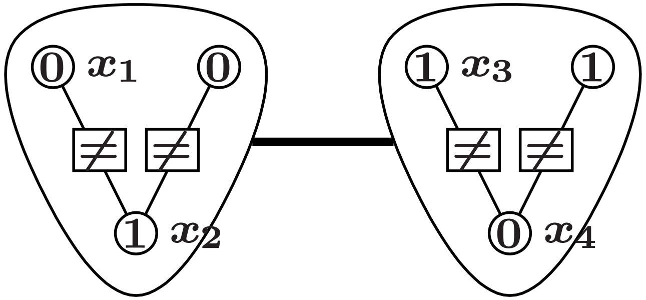


Hardness proof

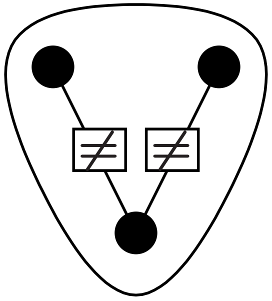
Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Suppose there is a counterexample to the fact that $R \in \Gamma$ is flip separable:

$(0, 1, 0, 1) \in R$
$(1, 0, 0, 1) \in R$
$(1, 0, 1, 0) \in R$
$(0, 1, 1, 0) \notin R$



We represent the edge by constraint $R(x_1, x_2, x_4, x_3)$.

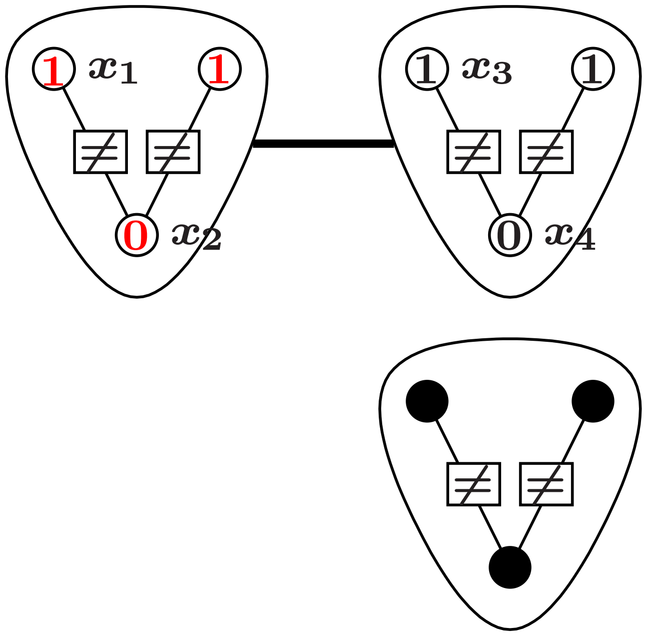


Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Suppose there is a counterexample to the fact that $R \in \Gamma$ is flip separable:

$$
\begin{aligned}
&(0,1,0,1) \in R \\
&(1,0,0,1) \in R \Leftarrow \\
&(1,0,1,0) \in R \\
&(0,1,1,0) \notin R
\end{aligned}
$$



We represent the edge by constraint $R(x_1, x_2, x_4, x_3)$.

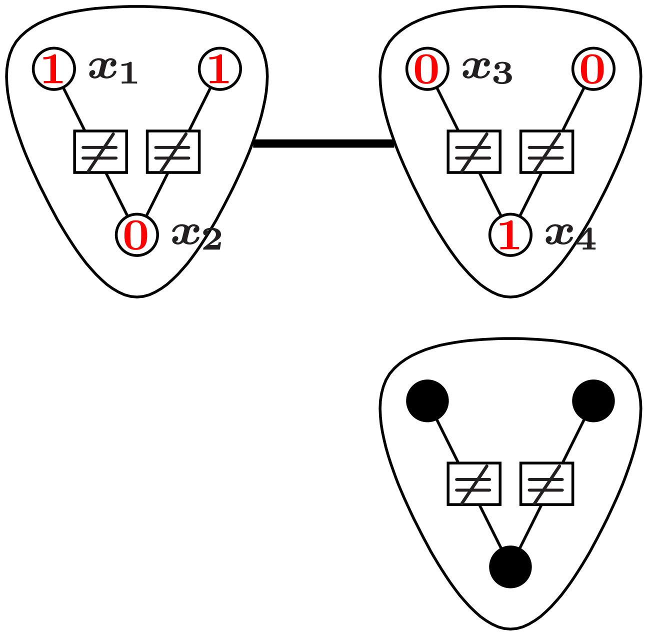
Flipping the first gadget is allowed...

Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Suppose there is a counterexample to the fact that $R \in \Gamma$ is flip separable:

$(0, 1, 0, 1) \in R$
$(1, 0, 0, 1) \in R$
$(1, 0, 1, 0) \in R \Leftarrow$
$(0, 1, 1, 0) \notin R$



We represent the edge by constraint $R(x_1, x_2, x_4, x_3)$.

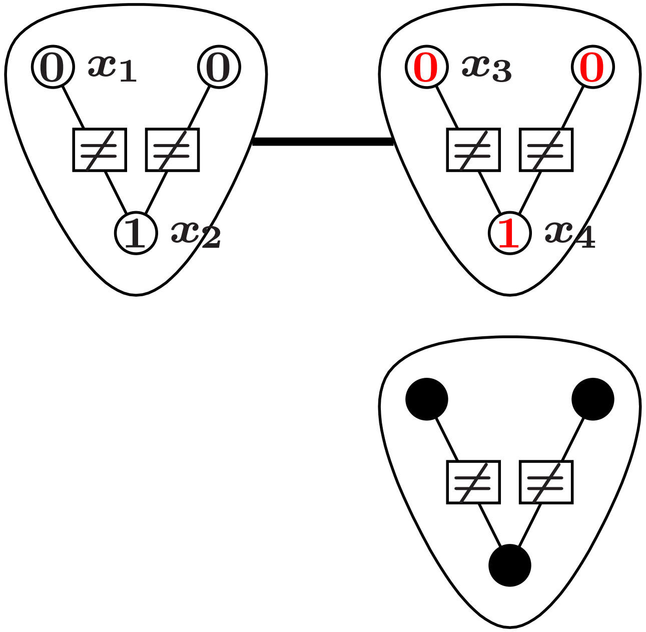
Flipping the first gadget is allowed...
Flipping both gadgets is allowed...

Hardness proof

Hardness proof: if Γ contains a relation that is not Horn and a relation that is not flip separable, then local search is W[1]-hard.

Suppose there is a counterexample to the fact that $R \in \Gamma$ is flip separable:

$(0,1,0,1) \in R$
$(1,0,0,1) \in R$
$(1,0,1,0) \in R$
$(0,1,1,0) \notin R \Leftarrow$



We represent the edge by constraint $R(x_1, x_2, x_4, x_3)$.

Flipping the first gadget is allowed...
Flipping both gadgets is allowed...
But second gadget cannot be flipped!

Main result

We have completed the complexity characterization of Γ-LOSE-WEIGHT:

Theorem: For every finite set Γ, Γ-LOSE-WEIGHT is fixed-parameter tractable if one of the following holds, and W[1]-hard otherwise:

- Every relation can be expressed by a Horn formula.
- Every relation is flip separable.

But something is strange...

Something strange

We have seen that local search is W[1]-hard for MINIMUM VERTEX COVER, even if the graph is bipartite.

Something strange

We have seen that local search is W[1]-hard for MINIMUM VERTEX COVER, even if the graph is bipartite.

$\Rightarrow$ But an optimum solution can be found in polynomial time!

Something strange

We have seen that local search is W[1]-hard for MINIMUM VERTEX COVER, even if the graph is bipartite.

$\Rightarrow$ But an optimum solution can be found in polynomial time!

The relation $x \vee y \vee \bar{z}$ is not Horn and not flip separable (for the tuple $(1, 0, 1)$, $\{2\}$ and $\{1, 2\}$ are flip sets but $\{1\}$ is not), thus local search is hard.

Something strange

We have seen that local search is W[1]-hard for MINIMUM VERTEX COVER, even if the graph is bipartite.

$\Rightarrow$ But an optimum solution can be found in polynomial time!

The relation $x \vee y \vee \bar{z}$ is not Horn and not flip separable (for the tuple $(1, 0, 1)$, $\{2\}$ and $\{1, 2\}$ are flip sets but $\{1\}$ is not), thus local search is hard.

$\Rightarrow$ But an optimum solution (all 0 assignment) can be found in polynomial time!

Something strange

We have seen that local search is W[1]-hard for MINIMUM VERTEX COVER, even if the graph is bipartite.

$\Rightarrow$ But an optimum solution can be found in polynomial time!

The relation $x \vee y \vee \bar{z}$ is not Horn and not flip separable (for the tuple $(1, 0, 1)$, $\{2\}$ and $\{1, 2\}$ are flip sets but $\{1\}$ is not), thus local search is hard.

$\Rightarrow$ But an optimum solution (all 0 assignment) can be found in polynomial time!

Counterintuitive results: finding a local improvement is hard, but finding the global optimum is easy.

We are answering the wrong question!

Strict vs. permissive

So far, we investigated **strict** local search algorithms:

Input:	A Γ-formula φ, a solution x for φ, and an integer k.
Task:	If there is a solution x' of φ with dist$(x, x') \leq k$ and weight$(x') <$ weight(x), then find such an x'.

Strict vs. permissive

So far, we investigated **strict** local search algorithms:

Input:	A Γ-formula φ, a solution x for φ, and an integer k.
Task:	If there is a solution x' of φ with $\text{dist}(x, x') \leq k$ and $\text{weight}(x') < \text{weight}(x)$, then find such an x'.

But a **permissive** local search algorithm would be equally useful:

Input:	A Γ-formula φ, a solution x for φ, and an integer k.
Task:	If there is a solution x' of φ with $\text{dist}(x, x') \leq k$ and $\text{weight}(x') < \text{weight}(x)$, then find any x'' with $\text{weight}(x'') < \text{weight}(x)$.

Our hardness result for strict local search does not rule out the possibility of a permissive algorithm.

Revised result

Theorem: For every finite set Γ, **strict** Γ-LOSE-WEIGHT is fixed-parameter tractable if one of the following holds, and W[1]-hard otherwise:

- Every relation can be expressed by a Horn formula.
- Every relation is flip separable.

Theorem: For every finite set Γ, **permissive** Γ-LOSE-WEIGHT is fixed-parameter tractable if one of the following holds, and W[1]-hard otherwise:

- Every relation can be expressed by a Horn formula.
- Every relation is flip separable.
- Every relation is 0-valid.

Conclusions

- Is it possible to efficiently search the local neighborhood?
- Parameterized complexity is the natural way to study.
- Might apply to YOUR problem as well!
- Schaefer-style classification for decreasing the weight of a solution in Boolean CSP.
- Main new definition: flip separable relations.
- Distinction between strict and permissive local search.