Multivariate Algorithmics for Voting

Britta Dorn

University of Ulm, Germany

FET'11

Voting systems

Multiple parties, different preferences → joint decision

- Political elections
- Group decisions: which restaurant/holiday destination/. . .
- Decisions about grants, job applicants
- Multi agent systems
- Aggregating results from several search engines
- Deciding which job to run first on a machine

Setting

An *election* consists of

- a set of candidates: 🟡, 🔴, 🔵, 🟢
- a set of votes (preference lists/rankings over candidates)

Voter 1 : 🟡 > 🔴 > 🔵 > 🟢
Voter 2 : 🟡 > 🔴 > 🔵 > 🟢
Voter 3 : 🔵 > 🟡 > 🔴 > 🟢
Voter 4 : 🟢 > 🔵 > 🔴 > 🟡
Voter 5 : 🔴 > 🟢 > 🔵 > 🟡

Problem 1: determine winner
Problem 2: determine consensus ranking ⟶ different voting rules

Efficient algorithms needed!

Setting

An *election* consists of

- a set of candidates: 🟡, 🔴, 🔵, 🟢
- a set of votes (preference lists/rankings over candidates)

Voter 1	:	🟡 > 🔴 > 🔵 > 🟢
Voter 2	:	🟡 > 🔴 > 🔵 > 🟢
Voter 3	:	🔵 > 🟡 > 🔴 > 🟢
Voter 4	:	🟢 > 🔵 > 🔴 > 🟡
Voter 5	:	🔴 > 🟢 > 🔵 > 🟡

Problem 1: determine winner
Problem 2: determine consensus ranking $\longrightarrow$ different voting rules

Efficient algorithms needed!
But for some voting rules, solving these problems is computationally hard. (Example: Kemeny ranking)

Dealing with computational hardness

Non-optimal/non-exact solution

- Approximation
- Heuristics
- Randomized algorithms

Optimal/exact solution

- Multivariate algorithmics

Dealing with computational hardness

Non-optimal/non-exact solution

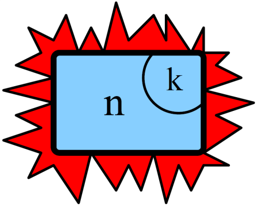
- Approximation
- Heuristics
- Randomized algorithms

Optimal/exact solution

- Multivariate algorithmics

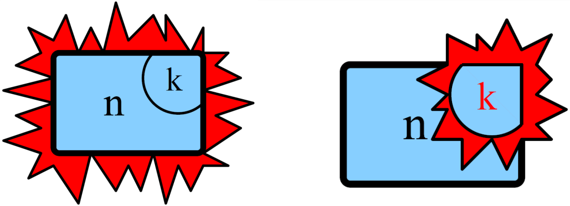
Multivariate algorithmics

NP-hard problems: exponential running time.



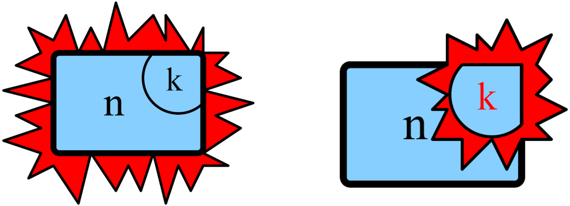
Multivariate algorithmics

NP-hard problems: exponential running time. But in some cases: The combinatorial explosion can be confined to a certain part of the input (parameter, k)



Multivariate algorithmics

NP-hard problems: exponential running time. But in some cases: The combinatorial explosion can be confined to a certain part of the input (parameter, *k*)



If the value of the parameter is small in certain settings: fast and optimal/exact solution possible!

Tasks part I

Task I

In hard cases: Investigate computational complexity of winner determination from a **multivariate algorithmic** point of view.

Natural parameters in voting problems:

- number of candidates
- number of voters
- amount of variation in voters' rankings
- distance of consensus ranking to voters' preference rankings

Example: Kemeny ranking becomes tractable if the number of candidates is small.

Tasks part II: The evil side

(Evil) ways to obtain preferred outcome of an election:

- Strategic voting
- Bribing
- Introducing/Deleting candidates or voters (control)
- Lobbying

Here: computational hardness consitutes a desired property!

Tasks part II: The evil side

- Strategic voting (manipulation)
- Bribing
- Introducing/Deleting candidates or voters (control)
- Lobbying

Good news

For most voting rules, the above problems are computationally hard.

Tasks part II: The evil side

- Strategic voting (manipulation)
- Bribing
- Introducing/Deleting candidates or voters (control)
- Lobbying

Good news

For most voting rules, the above problems are computationally hard.

Bad news

This doesn't mean that we are safe — it is still possible that they become tractable if certain parameters are small!
(E.g.: All of the above are tractable if the number of candidates is small)

Tasks part II: The evil side

- Strategic voting (manipulation)
- Bribing
- Introducing/Deleting candidates or voters (control)
- Lobbying

Good news

For most voting rules, the above problems are computationally hard.

Bad news

This doesn't mean that we are safe — it is still possible that they become tractable if certain parameters are small!
(E.g.: All of the above are tractable if the number of candidates is small)

Task II

Investigate the computational complexity of the above problems from a **multivariate algorithmic** point of view.

Summary

- Voting systems
- Two interesting kind of problems:
  1. Winner determination/consensus ranking
     → efficient algorithms wanted
  2. Manipulative actions:
     strategic voting, bribing, control, lobbying, . . .
     → computational hardness wanted
- Better insights and more fine-grained view by **multivariate algorithmics**