

Combining Metaheuristics with ILP Solvers: Construct, Merge, Solve & Adapt

Christian Blum

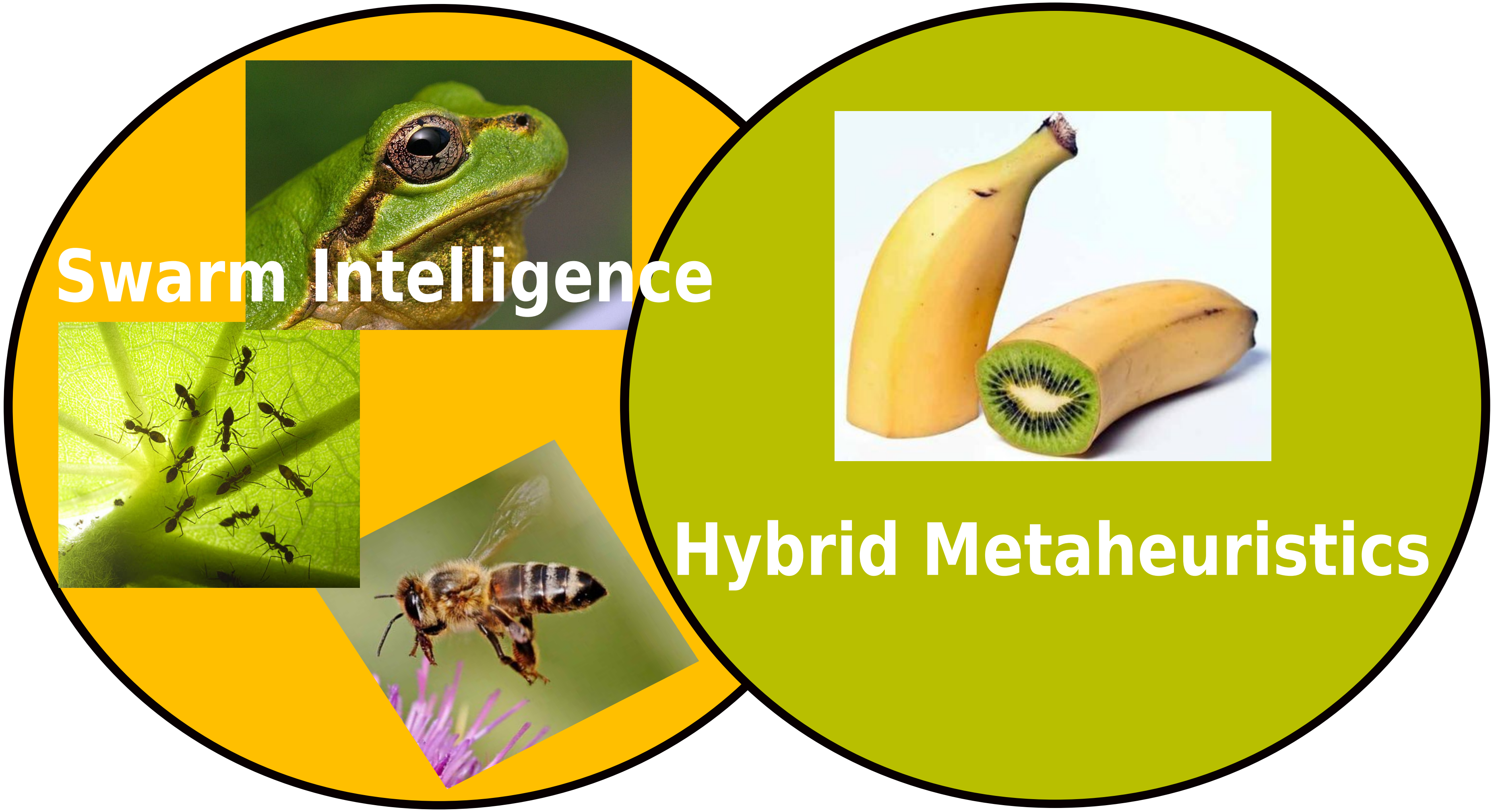
University Of The Basque Country

Ikerbasque, Basque Foundation For Science

ikerbasque
Basque Foundation for Science

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Research Topics in Recent Years



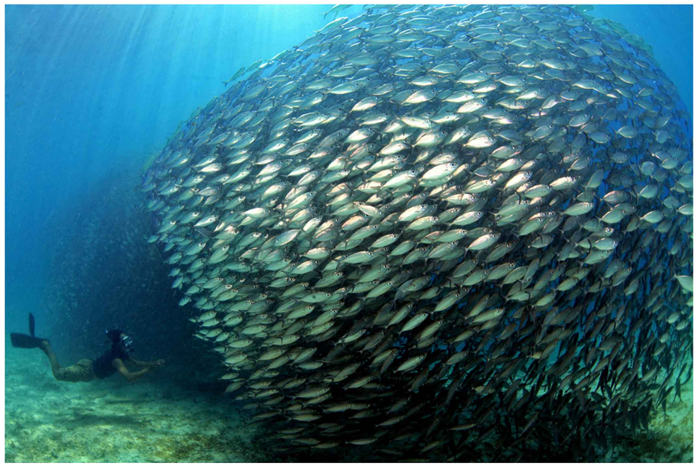
Lines of Research (1)

Swarm Intelligence

© Alex Wild (`http://www.myrmecos.net`)

What is swarm intelligence

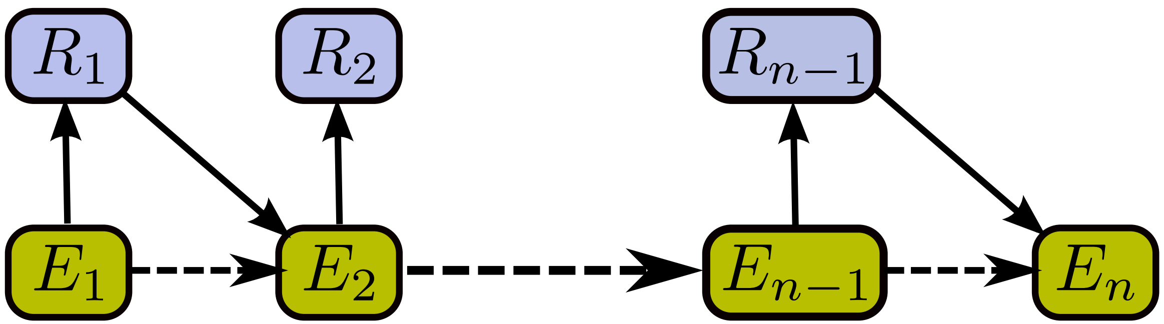
In a nutshell: AI discipline whose goal is designing intelligent multi-agent systems by taking inspiration from the collective behaviour of animal societies such as ant colonies, flocks of birds, or fish schools



Swarm intelligence

Properties:

- Consist of a set of simple entities
- Distributedness: No global control
- Self-organization by:
  - **Direct communication:** for example, by visual or chemical contact
  - **Indirect communication:** Stigmergy (Grassé, 1959)



Result: Complex tasks/behaviors can be accomplished/exhibited in cooperation

SI Topic 1: Self-Synchronized Duty-Cycling in Sensor Networks

Inspiration: Self-synchronized activity phases of ant colonies

SI Topic 1: Self-Synchronized Duty-Cycling

Biologist discovered:

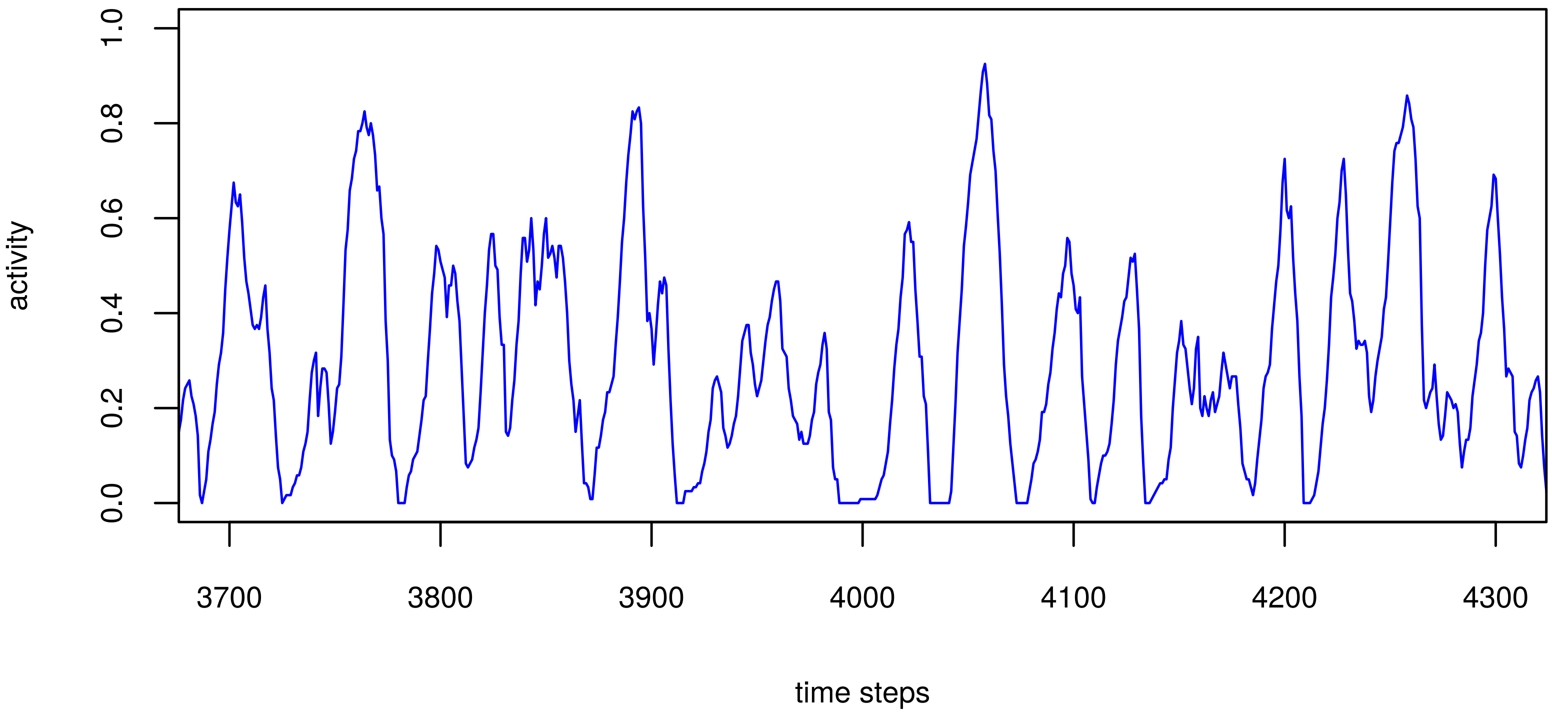
- Colonies of ants show synchronized activity patterns
- Synchronization is achieved in a self-organized way: self-synchronization
- Synchronized activity ...
  1. ... provides a mechanism for information propagation
  2. ... facilitates the sampling of information from other individuals

Mathematical model:

J. Delgado and R.V. Solé. **Self-synchronization and task fulfilment in ant colonies**, *Journal of Theoretical Biology*, 205, 433–441 (2000)

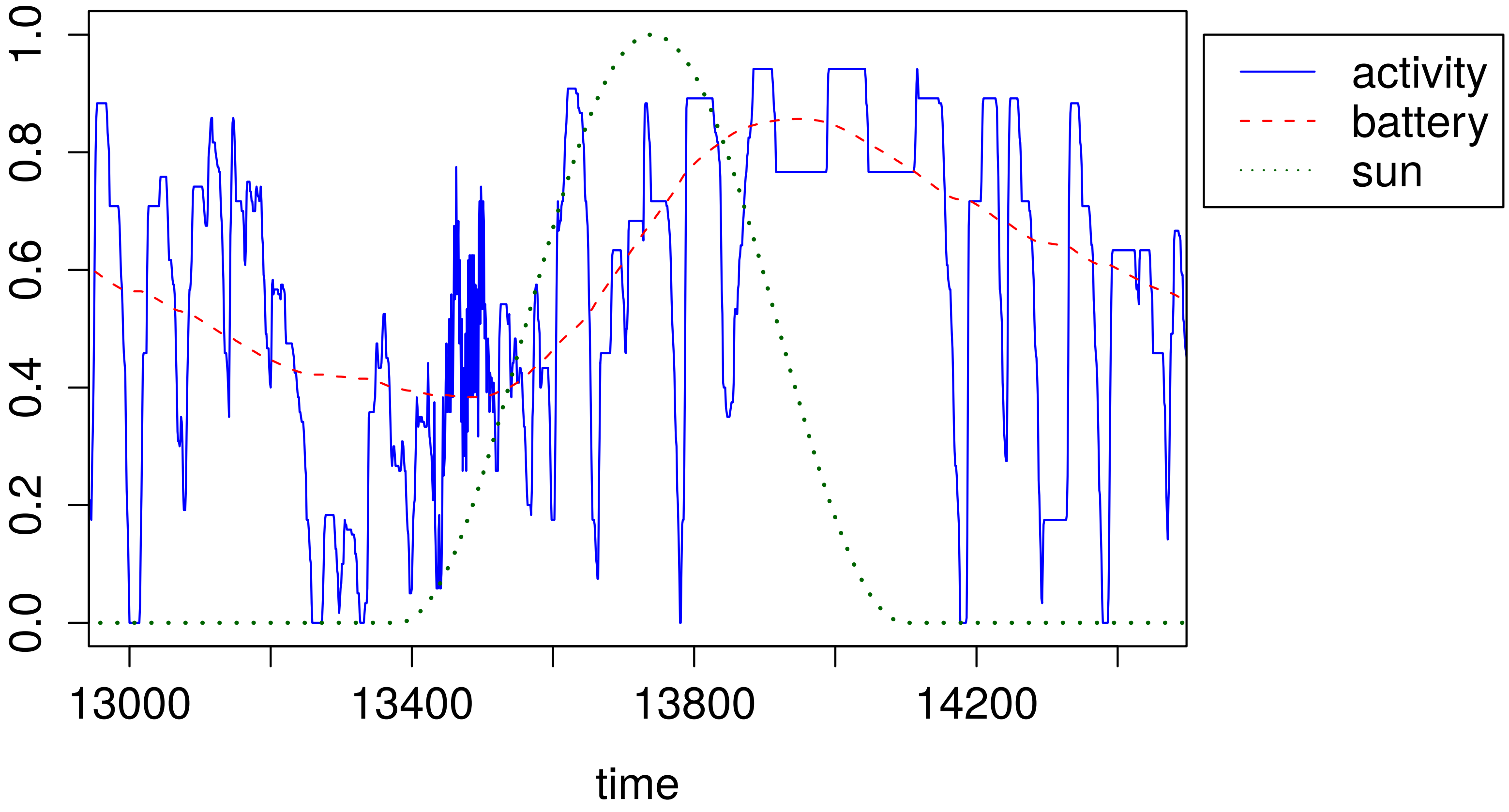
SI Topic 1: Self-Synchronized Duty-Cycling

Graphic: Mean activity of an ant colony over time



Self-Synchronized Duty-Cycling: simulation

Example: Behaviour in simulator Shawn



Advantages: Completely self-organized, adaptive, and robust against packet loss

Self-Synchronized Duty-Cycling: papers

Representative papers:

- H. Hernández and C. Blum. **Foundations of ANTCYCLE: Self-synchronized duty-cycling in mobile sensor networks**. *The Computer Journal*, 2011.

- H. Hernández et al. **A protocol for self-synchronized duty-cycling in sensor networks: Generic implementation in WISELIB**. *Proceedings of the 6th International Conference on Mobile Ad-hoc and Sensor Networks*, IEEE Press, 2010.

SI Topic 2: Distributed Problem Solving in Wireless Ad-hoc Networks

Inspiration: Self-desynchronization of Japanese tree frogs

SI Topic 2: Distributed Problem Solving

Biologist discovered:

- Male Japanese Tree Frogs de-couple their calls
- Why?
  - ⋆ The purpose of the calls is to attract females
  - ⋆ Female frogs cannot distinguish calls close in time
  - ⋆ Result: females cannot determine the location of males

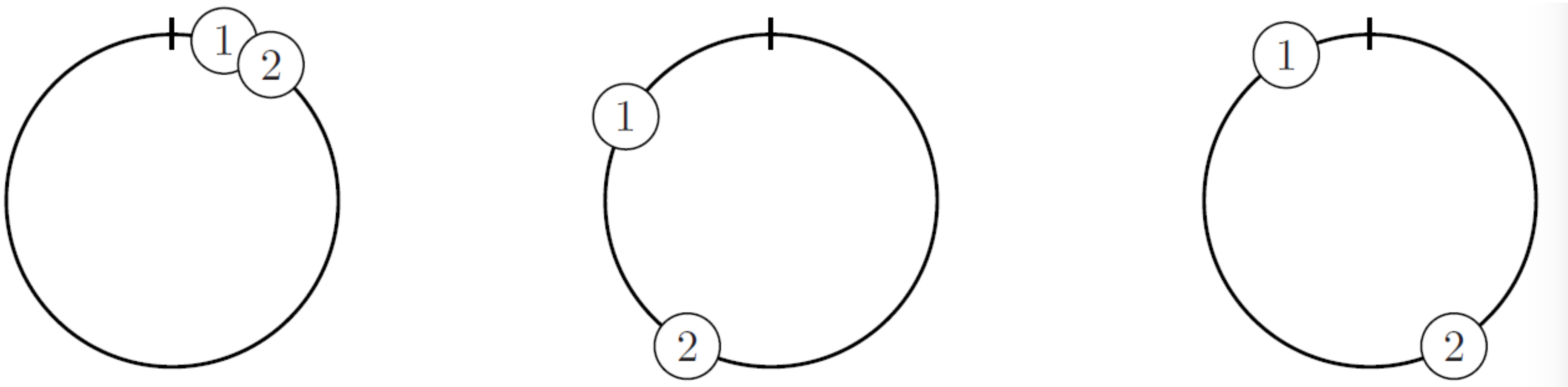
Mathematical model:

I. Aihara, H. Kitahata, K. Yoshikawa and K. Aihara. **Mathematical modeling of frogs' calling behavior and its possible applications to artificial life and robotics.** *Artificial Life and Robotics*, 12(1):29–32, 2008.

SI Topic 2: Distributed Problem Solving

Model components:

- A set of pulse-coupled oscillators.
- Some oscillators are coupled, others are independent of each other
- Each oscillator i has a phase $\theta_i \in [0, 1)$ which changes over time



Distributed Problem Solving: papers

Representative papers:

- H. Hernández and C. Blum. **Distributed Graph Coloring: An Approach Based on the Calling Behavior of Japanese Tree Frogs**. *Swarm Intelligence*, 2012.
- C. Blum, B. Calvo, M. J. Blesa. **FrogCOL and FrogMIS: new decentralized algorithms for finding large independent sets in graphs**. *Swarm Intelligence*, 2015.

Award: **Best Paper Award**

- H. Hernández and C. Blum. **Distributed graph coloring in wireless ad hoc networks: A light-weight algorithm based on Japanese tree frogs' calling behaviour**. *Wireless Mobile Networking Conference 2011*.

Swarm Intelligence: Quo vadis?

- Problem: Swarm intelligence has attracted too many people
- As a consequence:
  1. Experienced researchers were overwhelmed with reviewing
  2. People who should have never been asked to do so did reviewing work
- Therefore: nowadays we find numerous papers in the literature that are either
  1. Non-sense, or
  2. Re-inventing the wheel

First steps against this trend:

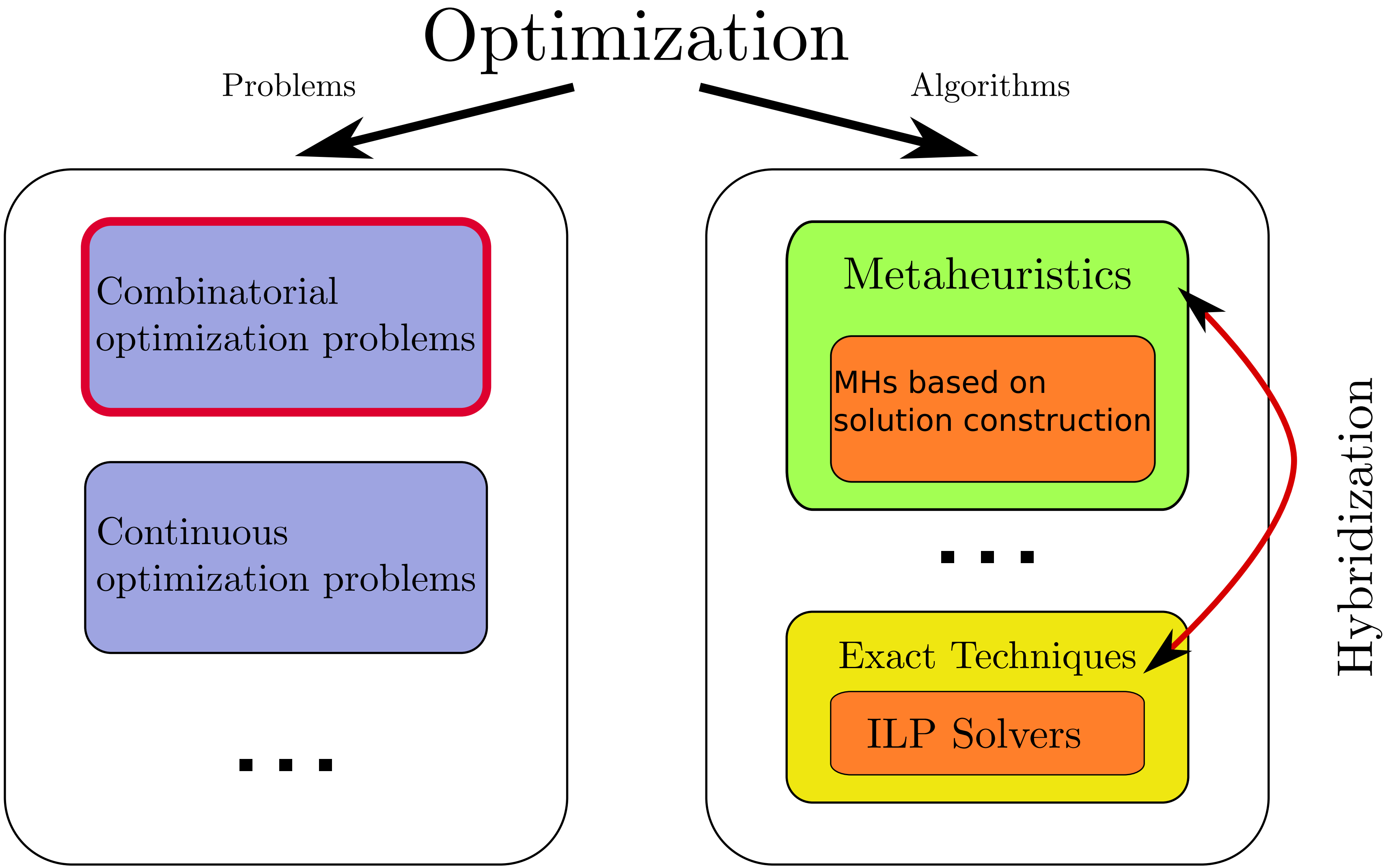
- Some journals (J. of Heur. , Comp. & Oper. Res.) ask for algorithms to be described in metahpor-free language
- Colleagues start to expose the problem (G. Rudolph , K. Sörensen)

Lines of Research (2)

Hybrid Metaheuristics

© www.hemmy.net

Preliminaries: Preparing the Grounds



Hybrid metaheuristics: definition

Definition: What is a hybrid metaheuristic?

- Problem: a precise definition is not possible/desirable

Possible characterization:

A technique that results from the combination of a metaheuristic with other techniques for optimization

What is meant by: other techniques for optimization ?

- Metaheuristics
- Branch & bound
- Dynamic programming
- Integer Linear Programming (ILP) techniques

Hybrid metaheuristics: history

History:

- For a long time the different communities co-existed quite isolated
- Hybrid approaches were developed already early, but only sporadically
- Only since about 15 years the published body of research grows significantly:
  1. 1999: CP-AI-OR Conferences/Workshops
  2. 2004: Workshop series on Hybrid Metaheuristics (HM 200X)
  3. 2006: Matheuristics Workshops

Consequence: The term hybrid metaheuristics identifies a new line of research

Motivation behind my work on hybrid metaheuristics

- In the field of metaheuristics we have rules of thumb:
  1. If, for your problem, there is a **good greedy heuristic** apply GRASP or Iterated Greedy
  2. If, for your problem, there is an **efficient neighborhood** apply Iterated Local Search or Tabu Search
- In contrast, for hybrid metaheuristics not much is known
  - ⋆ We only have very few generally applicable techniques
  - ⋆ We do not really know for which type of problem they work well
- Disadvantage of mathematical programming: Considerable amount of expert knowledge necessary to implement a well-working technique
- Goal: take profit from general purpose ILP solvers within metaheuristics

Construct, Merge, Solve & Adapt (CMSA)

Short description

Why combining metaheuristics with ILP Solvers?

General advantage of metaheuristics:

- Very good in exploiting information on the problem (greedy heuristics)
- Generally very good in obtaining high-quality solutions for medium and even large size problem instances

However:

- Metaheuristics may also reach their limits with growing problem instance size
- Metaheuristics fail when the information on the problem is misleading

Goal: Taking profit from valuable optimization expertise that went into the development of ILP solvers even in the context of large problem instances

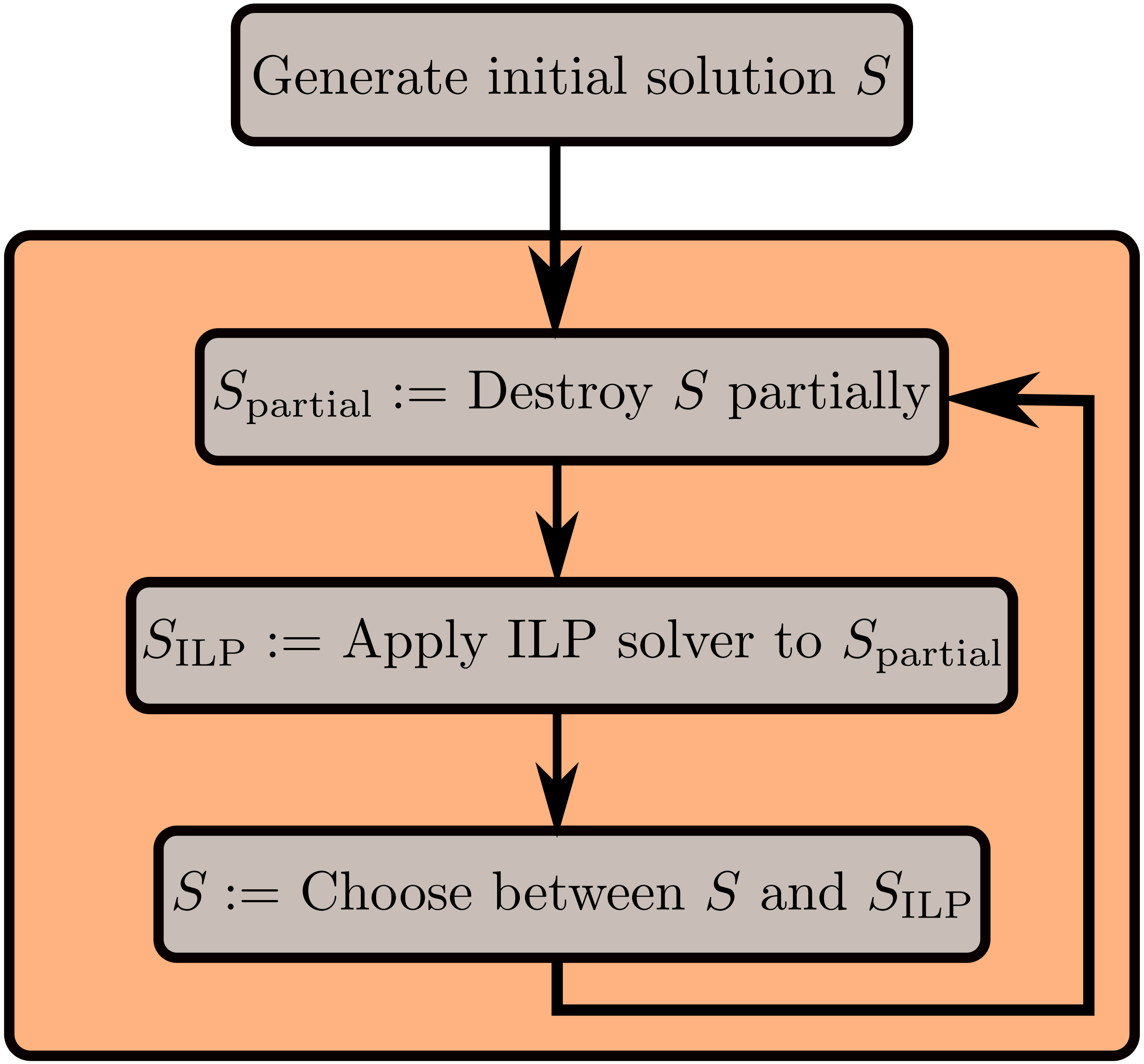
Standard: Large Neighborhood Search

- Small neighborhoods:
  1. **Advantage:** It is fast to find an improving neighbor (if any)
  2. **Disadvantage:** The average quality of the local minima is low
- Large neighborhoods:
  1. **Advantage:** The average quality of the local minima is high
  2. **Disadvantage:** Finding an improving neighbor might itself be NP-hard due to the size of the neigbhorhood

Ways of examining large neighborhoods:

- Heuristically
- Exact techniques: for example an ILP solver

ILP-based large neighborhood search: ILP-LNS



Hypothesis and resulting research question

In our experience: LNS works especially well when

1. The **number of solution components** (variables) is is not high
2. The **number of components in a solution** is not too small

Question:

What kind of general algorithm can we apply when the above conditions are not fullfilled?

Construct, Merge, Solve & Adapt: Principal Idea

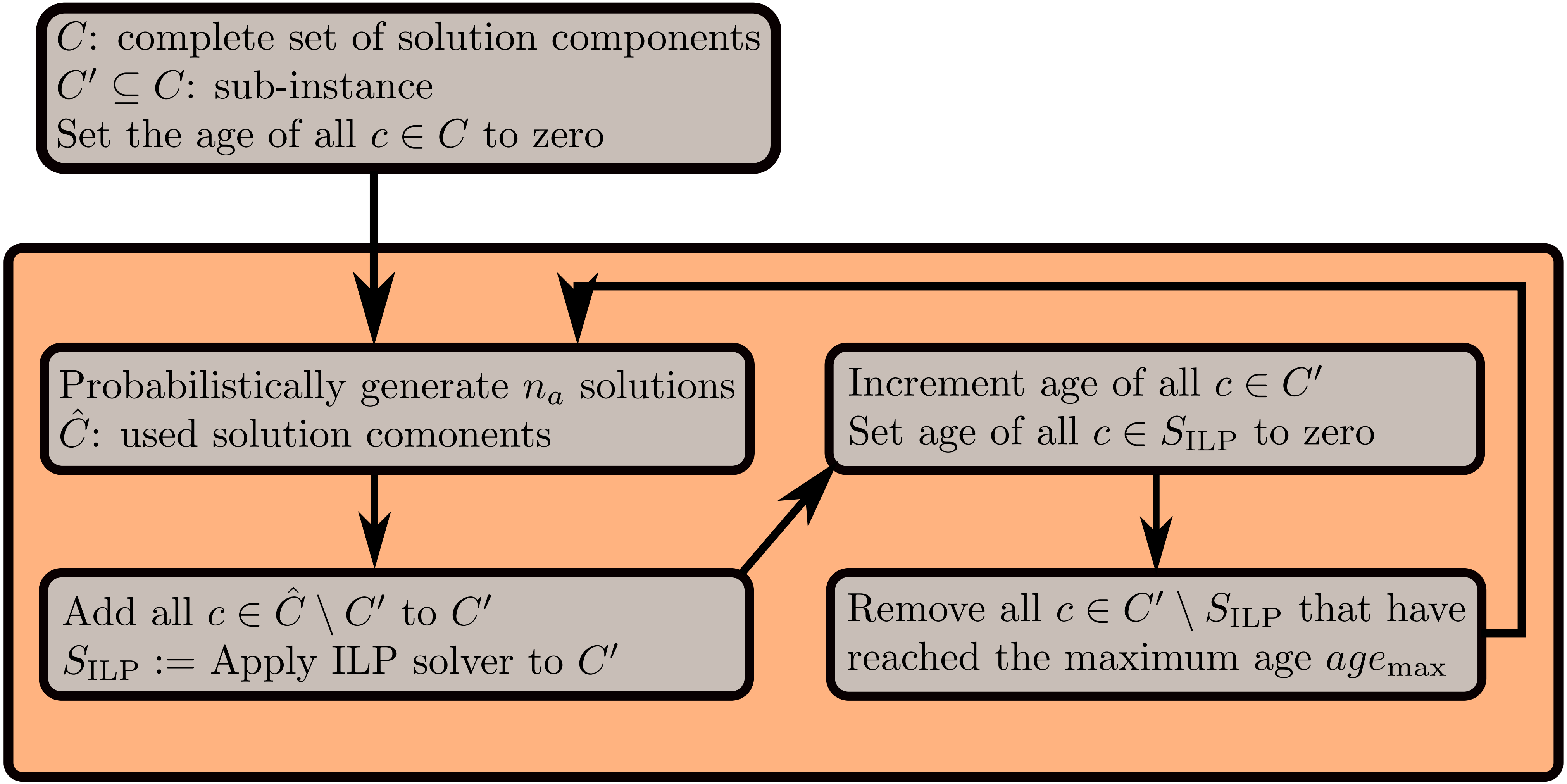
Observation: In the presence of a large number of solutions components, many of them only lead to bad solutions

Idea: Exclude the presumably bad solution components from the ILP

Steps of the proposed method:

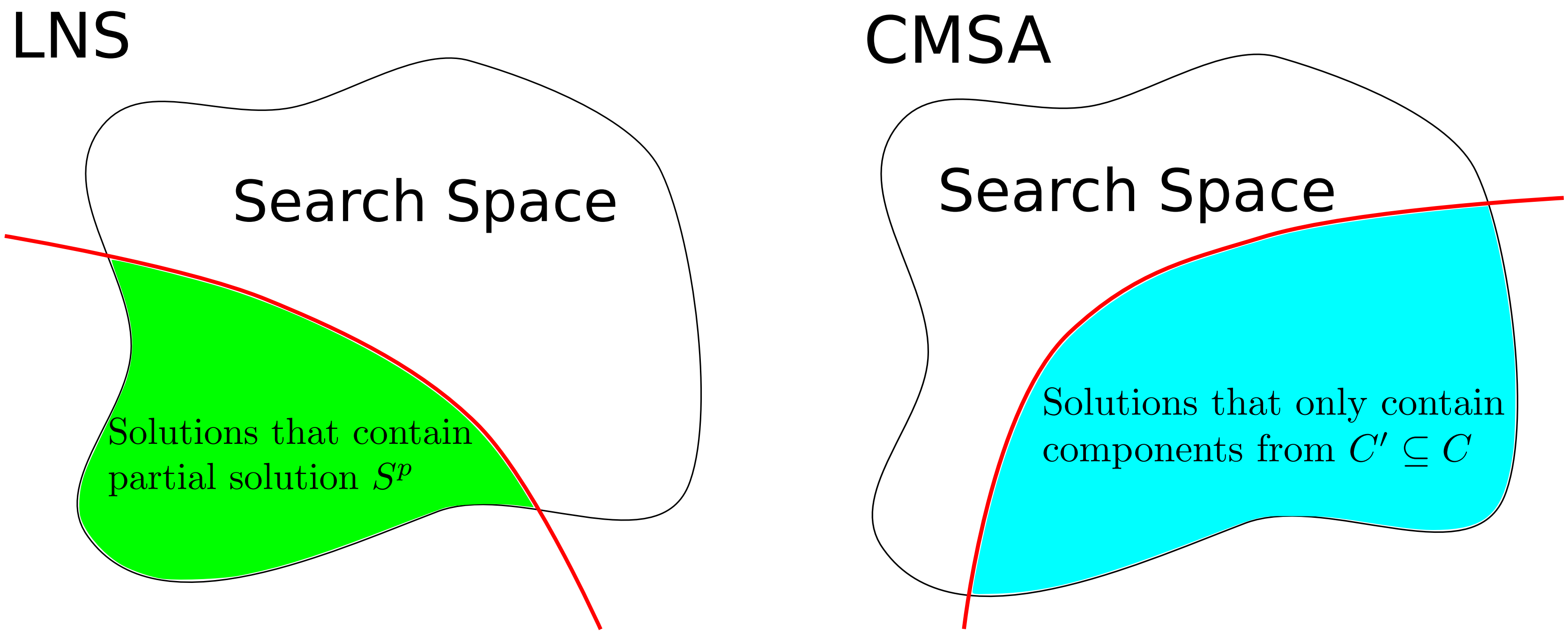
- Iteratively generate presumably good solutions in a probabilistic way
- Assemble a sub-instance from the used solution components
- Solve the sub-instance by means of an ILP solver
- Delete useless solution components from the sub-instance

Construct, Merge, Solve & Adapt: Flow Diagram



Differences between LNS and CMSA: summarized

How is the original problem instance reduced?



How is the sub-instance of the next iteration generated?

- LNS: Partial destruction of the incumbent solution
- CMSA: Generating new solutions and removing **old** solution components

Longest common subsequence (LCS) problem (1)

Notation: What is a subsequence of a string?

A string t is called a subsequence of a string x,
iff t can be produced from x by deleting characters

Example: Is AAT a subsequence of ACAGTTA?

ACAGTTA

Longest common subsequence (LCS) problem (2)

Problem definition (restricted to two input sequence)

Given: A problem instance (x, y, Σ), where

- x and y are input sequences over the alphabet Σ

Optimization goal:

Find a longest string t^* that is a subsequence of strings x and y $\rightarrow$ a longest common subsequence

Repetition-free longest common subsequence problem

- Restriction: No letter **may appear more than once** in a valid solution
- Proposed in: 2010 in *Discrete Applied Mathematics*
- Hardness: APX-hard (shown in above paper)
- Motivation: Genome rearrangement where duplicate genes are basically not considered
- Existing algorithms:
  1. Three simple heuristics, *Discrete Applied Mathematics*, 2010
  2. An Evolutionary Algorithm, *Operations Research Letters*, 2013

A simple constructive RFLCS heuristic: Best-Next (1)

Principle: Builds a solution sequentially from left to right

1: **input:** a problem instance (x, y, Σ)
2: **initialization:** $t := \epsilon$ (where ϵ is the empty string)
3: **while** $|\Sigma_t^{\mathrm{nd}}| > 0$ **do**
4: $\quad a := \mathsf{ChooseFrom}(\Sigma_t^{\mathrm{nd}})$
5: $\quad t := ta$
6: **end while**
7: **output:** a repetition-free common subsequence t

Question: How is Σ_t^{nd} defined?

A simple constructive LCS heuristic: Best-Next (2)

Example: Given is

- Problem instance $(x, y, \Sigma = \{A, C, T, G\})$ where
  - ⋆ $x = \text{ATCTAGCTG}$
  - ⋆ $y = \text{TACCATGTG}$
- Partial solution $t = \text{AC}$

$$\overbrace{\underline{A}\ \ T\ \ \underset{\uparrow\, p_x}{\underline{C}}}^{x^+}\quad \overbrace{\underset{\uparrow\, p_x^{T}}{T}\ \ A\ \ \underset{\uparrow\, p_x^{G}}{G}\ \ C\ \ T\ \ G}^{x^-}\qquad\qquad \overbrace{T\ \ \underline{A}\ \ \underset{\uparrow\, p_y}{\underline{C}}}^{y^+}\quad \overbrace{C\ \ A\ \ \underset{\uparrow\, p_y^{T}}{T}\ \ \underset{\uparrow\, p_y^{G}}{G}\ \ T\ \ G}^{y^-}$$

Result: $\Sigma_t^{\text{nd}} = \{\text{T}\}$

Greedy function

$\boxed{\text{Greedy function:}}$

$$\eta(ta) := \left(\frac{p_x^a - p_x}{|x^-|} + \frac{p_y^a - p_y}{|y^-|} \right)^{-1}, \quad \forall a \in \Sigma_t^{\mathrm{nd}}$$

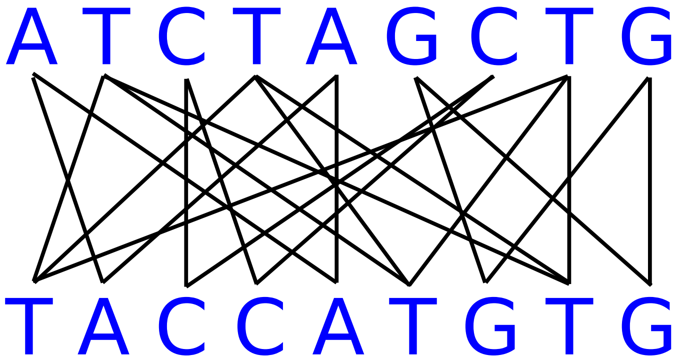
$$\overbrace{\underline{\mathrm{A}}\ \mathrm{T}\ \underset{\overset{\uparrow}{p_x}}{\underline{\mathrm{C}}}}^{x^+}\ \ \overbrace{\underset{\overset{\uparrow}{p_x^{\mathrm{T}}}}{\mathrm{T}}\ \mathrm{A}\ \underset{\overset{\uparrow}{p_x^{\mathrm{G}}}}{\mathrm{G}}\ \mathrm{C}\ \mathrm{T}\ \mathrm{G}}^{x^-} \qquad \overbrace{\mathrm{T}\ \underline{\mathrm{A}}\ \underset{\overset{\uparrow}{p_y}}{\underline{\mathrm{C}}}}^{y^+}\ \ \overbrace{\mathrm{C}\ \mathrm{A}\ \underset{\overset{\uparrow}{p_y^{\mathrm{T}}}}{\mathrm{T}}\ \underset{\overset{\uparrow}{p_y^{\mathrm{G}}}}{\mathrm{G}}\ \mathrm{T}\ \mathrm{G}}^{y^-}$$

ILP Model (1)

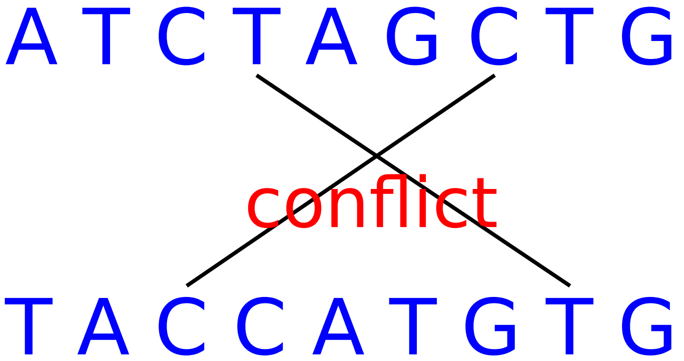
Set of binary variables:

For each position i of x and j of y such that $x[i] = y[j]$ the model has a variable $z_{i,j}$

Example set of variables



Example of a conflict



ILP Model (2)

$$\textbf{max} \sum_{z_{i,j} \in Z} z_{i,j} \tag{1}$$

subject to:

$$\sum_{z_{i,j} \in Z_a} z_{i,j} \leq 1 \quad \text{for } a \in \Sigma \tag{2}$$

$$z_{i,j} + z_{k,l} \leq 1 \quad \text{for all } z_{i,j} \text{ and } z_{k,l} \text{ being in conflict} \tag{3}$$

$$z_{i,j} \in \{0, 1\} \quad \text{for } z_{i,j} \in Z \tag{4}$$

Hereby:

- $z_{i,j} \in Z_a$ iff $x[i] = y[j] = a$
- $z_{i,j}$ and $z_{k,l}$ are in conflict iff $i < k$ and $j > l$ OR $i > k$ and $j < l$

Experimental evaluation: benchmark instances

Set1: 30 instances for each combination of

- Input sequence length: $n \in \{32, 64, 128, 256, 512, 1024, 2028, 4048\}$
- Alphabet size: $|\Sigma| \in \{n/8, n/4, 3n/8, n/2, 5n/8, 3n/4, 7n/8\}$

Set2: 30 instances for each combination of

- Alphabet size: $|\Sigma| \in \{4, 8, 16, 32, 64, 128, 256, 512\}$
- Maximal number of repetitions of each letter: $rep \in \{3, 4, 5, 6, 7, 8\}$

Tuning: CMSA's parameters are tuned by irace for each alphabet size

Experimental results: performance of CPLEX

Set1:

- Input sequence length: $n \in \{\mathbf{32}, \mathbf{64}, \mathbf{128}, \mathbf{256}, \mathbf{512}, 1024, 2028, 4048\}$
- Alphabet size: $|\Sigma| \in \{\mathbf{n/8}, \mathbf{n/4}, \mathbf{3n/8}, \mathbf{n/2}, 5n/8, 3n/4, 7n/8\}$

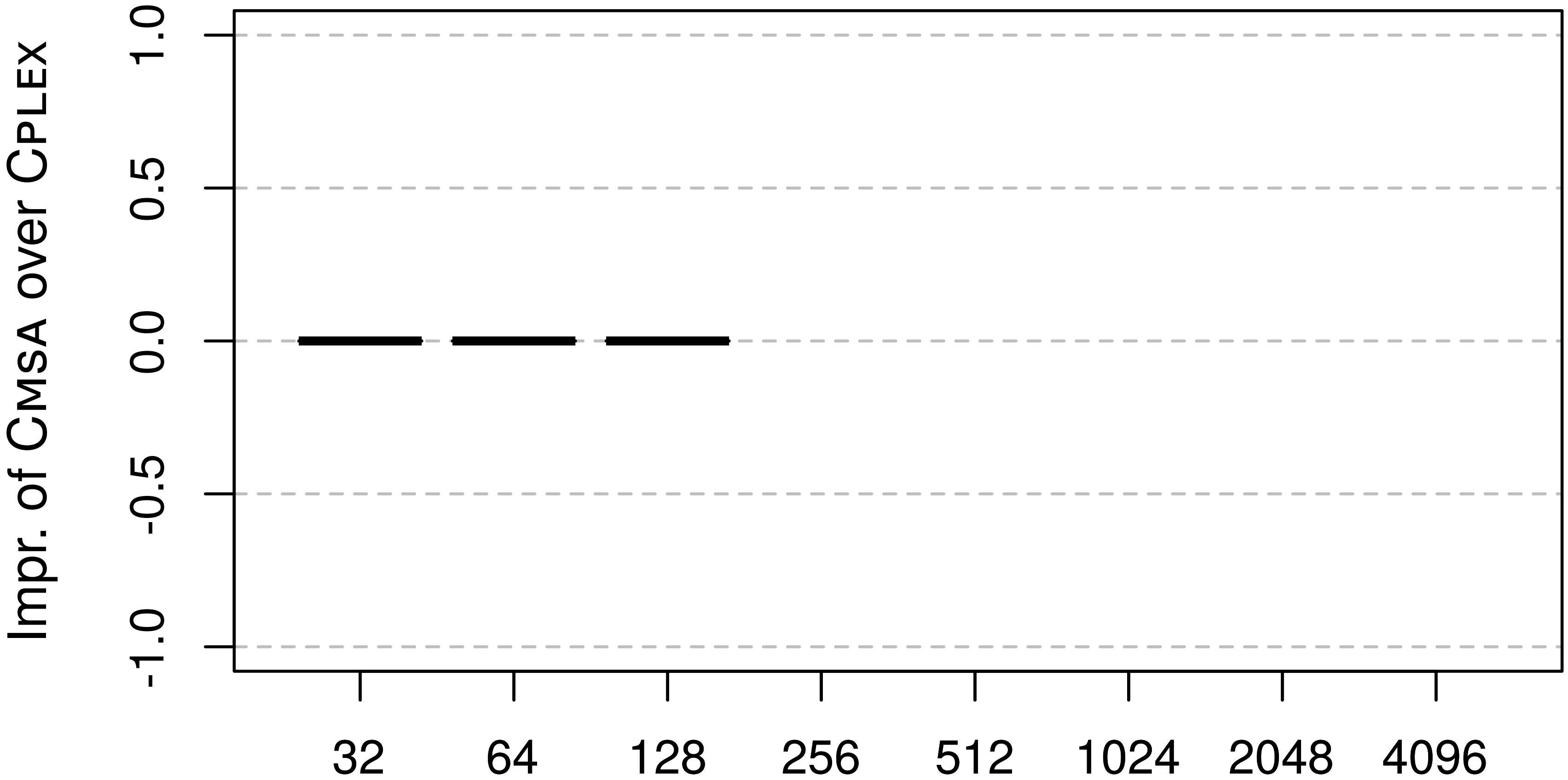
Set2:

- Alphabet size: $|\Sigma| \in \{\mathbf{4}, \mathbf{8}, \mathbf{16}, \mathbf{32}, \mathbf{64}, 128, 256, 512\}$
- Maximal number of repetitions of each letter: $rep \in \{3, 4, 5, \mathbf{6}, \mathbf{7}, \mathbf{8}\}$

Result: CPLEX is able to solve nearly all exisiting problem instances from the literature to optimality

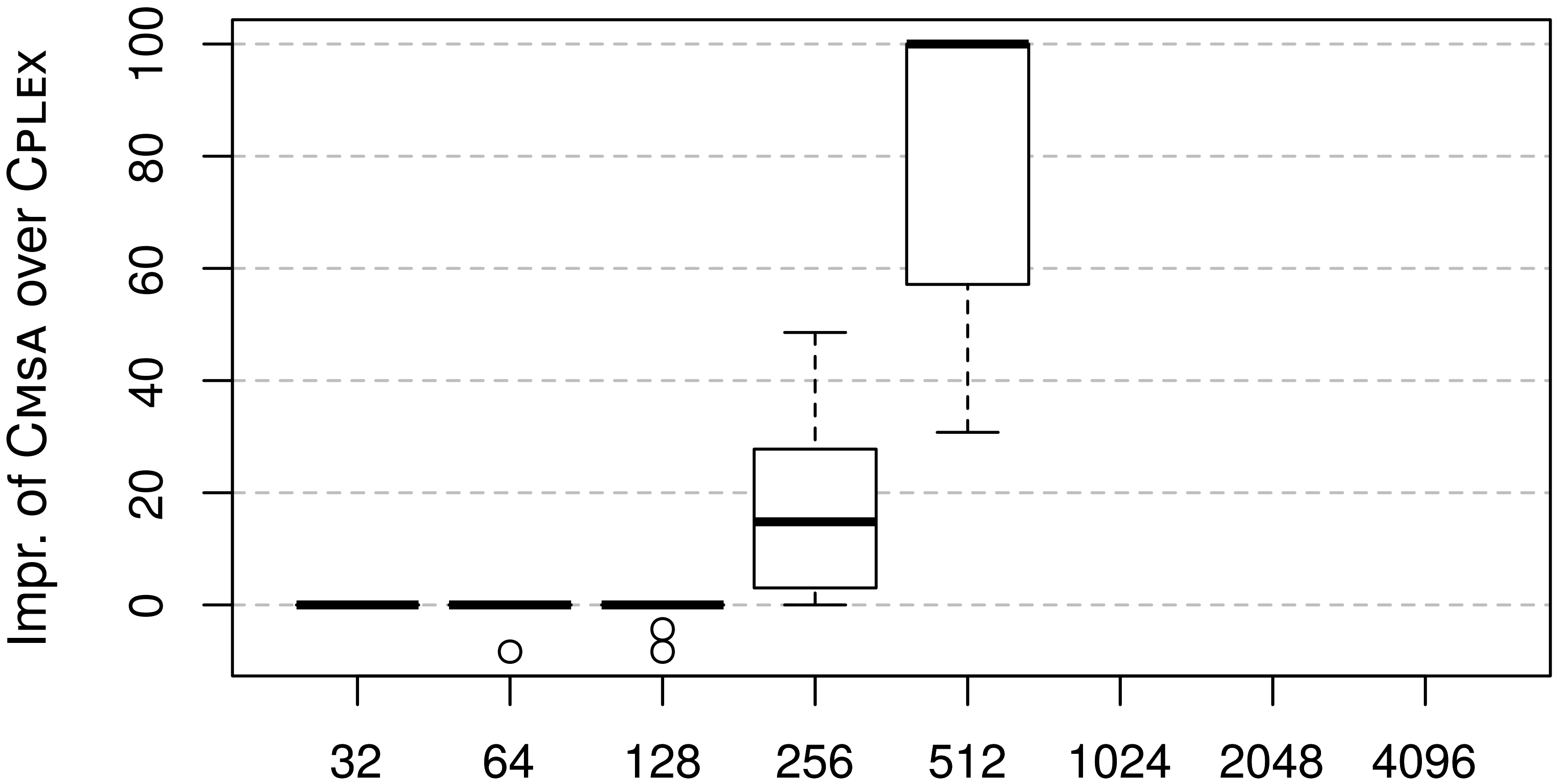
Experimental results: Set1

Improvement of CMSA over CPLEX: alphabet size $n/8$



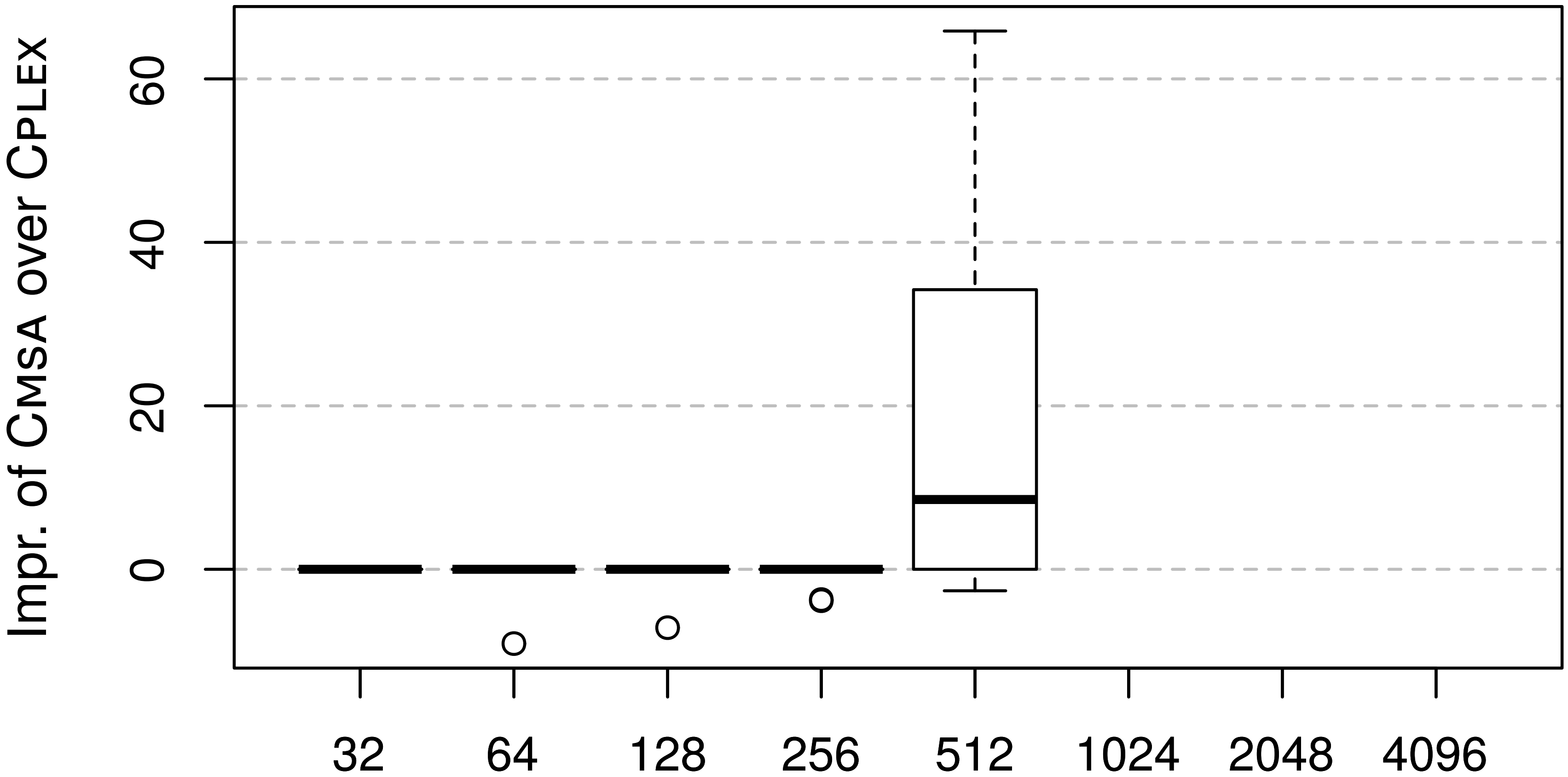
Experimental results: Set1

Improvement of CMSA over CPLEX: alphabet size $n/2$



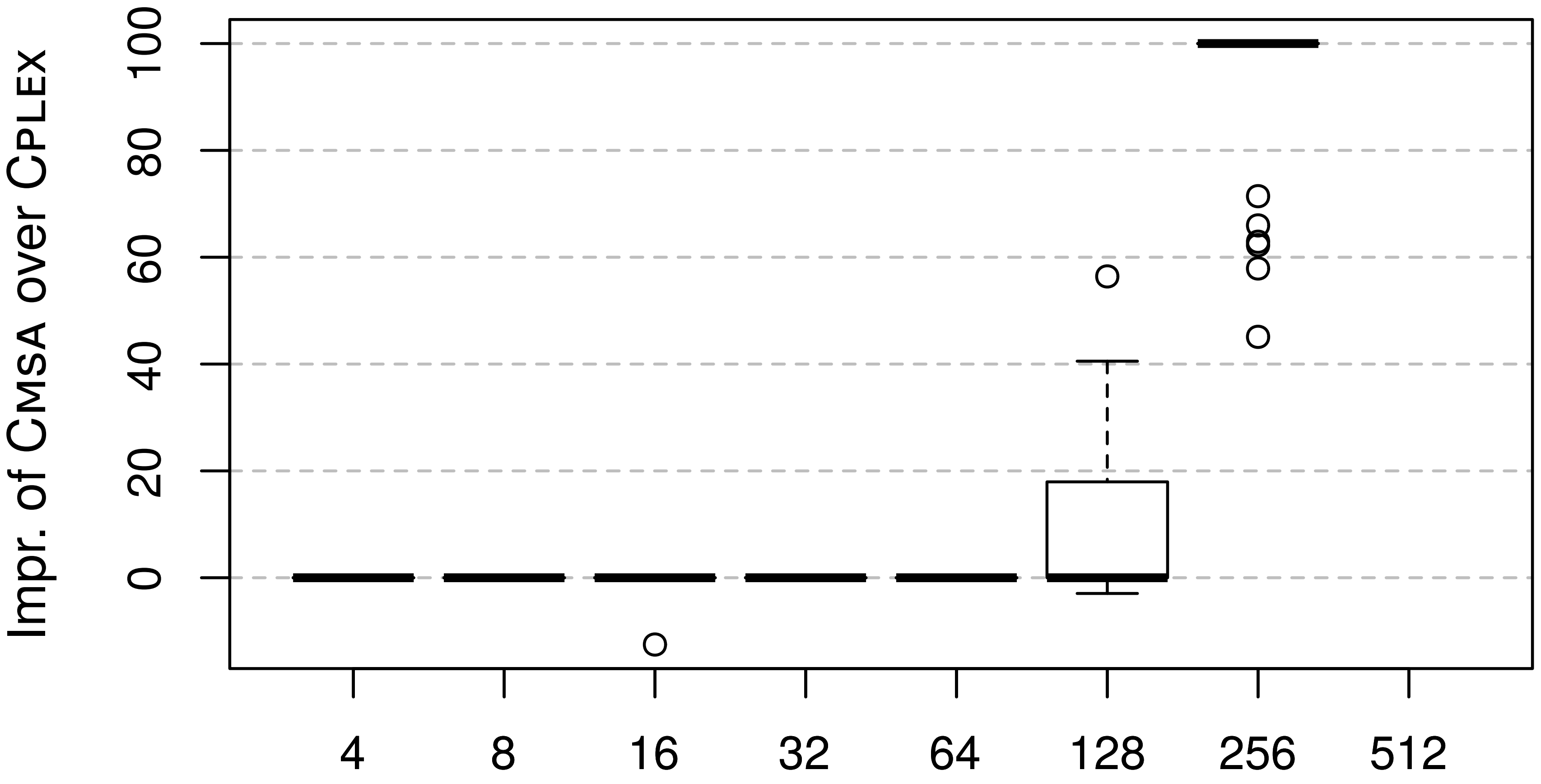
Experimental results: Set1

Improvement of CMSA over CPLEX: alphabet size $7n/8$



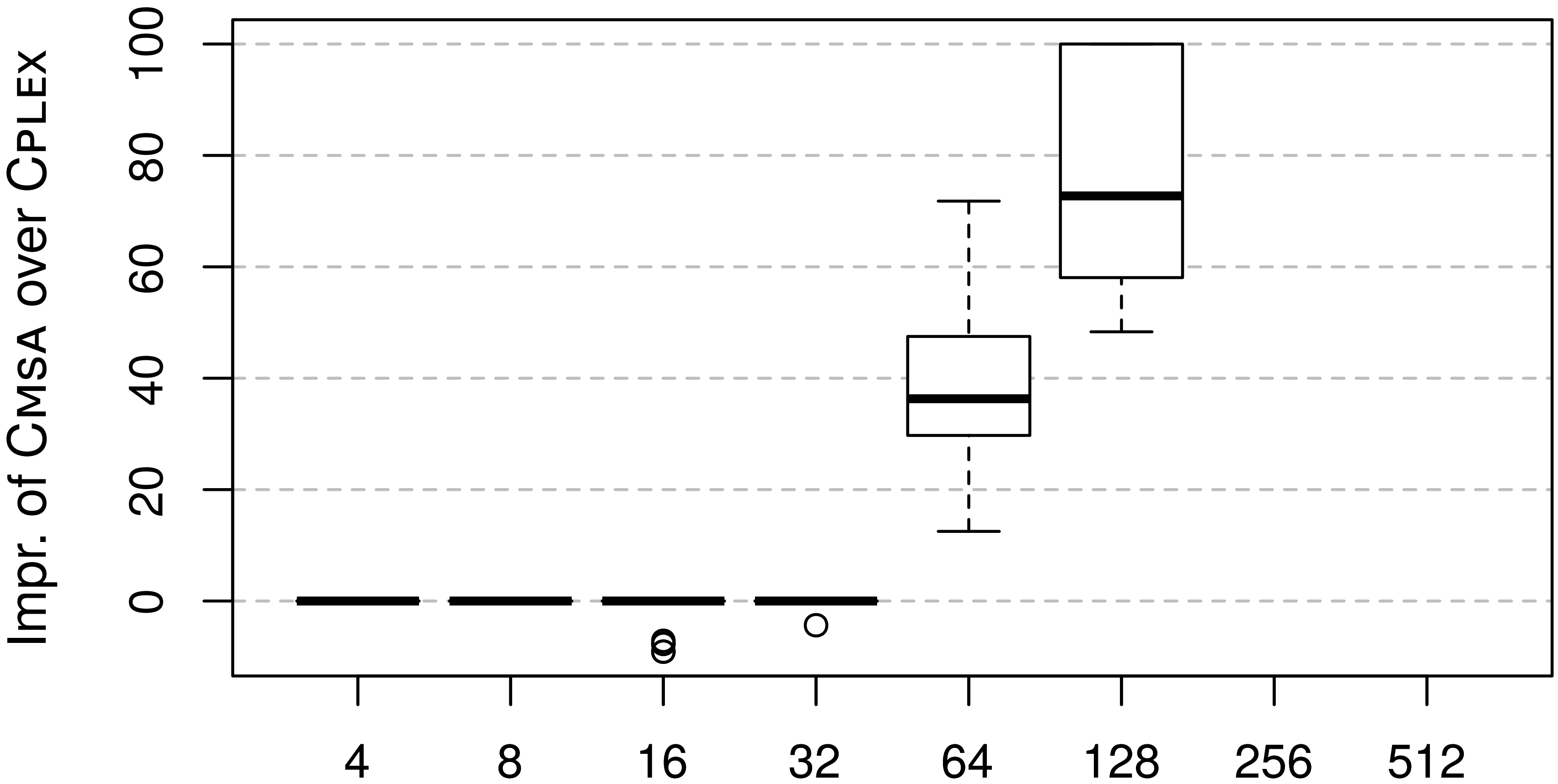
Experimental results: Set2

Improvement of CMSA over CPLEX: 3 reps



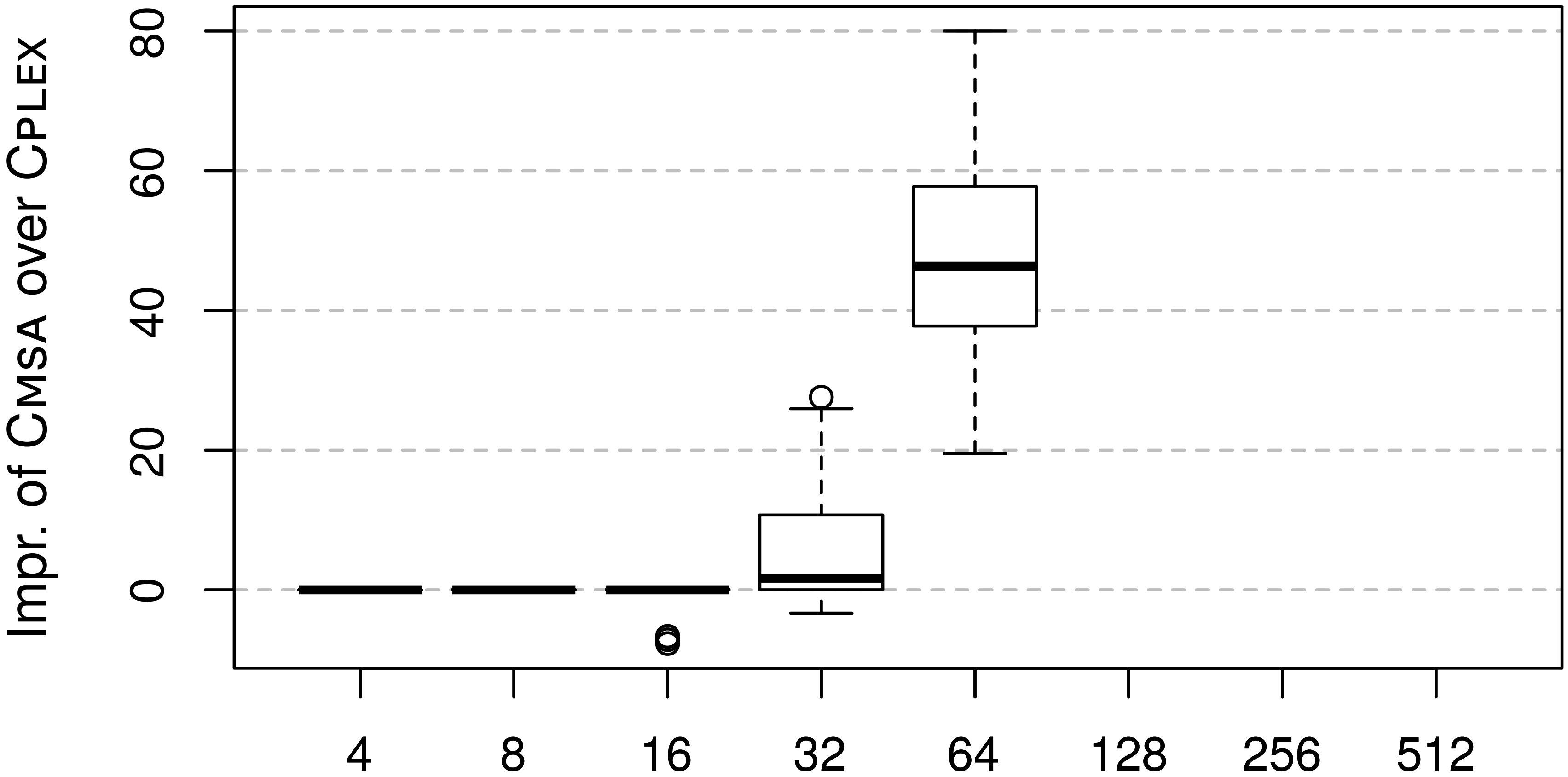
Experimental results: Set2

Improvement of CMSA over CPLEX: 6 reps

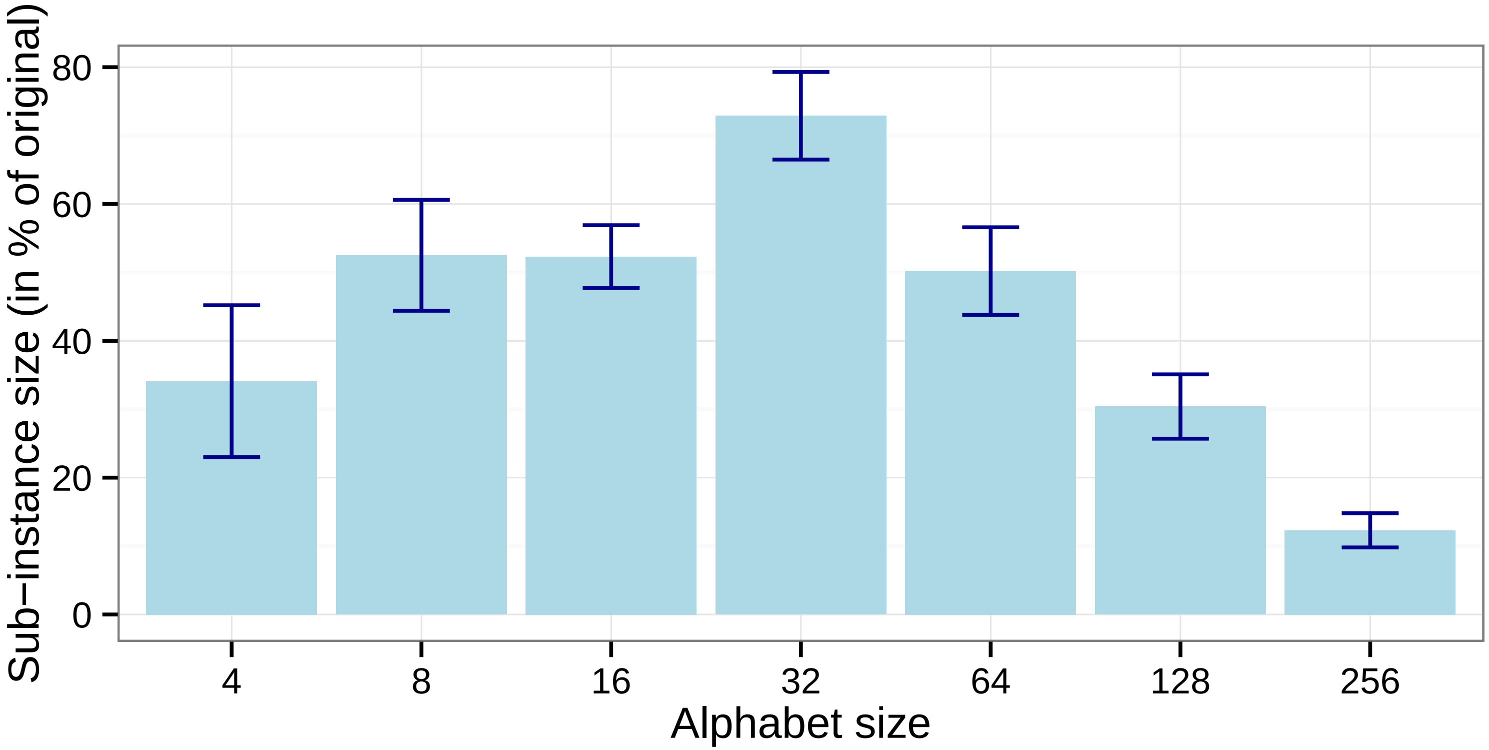


Experimental results: Set2

Improvement of CMSA over CPLEX: 8 reps



Experimental results: size of sub-instances



Relation between LNS and CMSA

First experimental study

Reminder: Intuition

- CMSA will have advantages over LNS when solutions are small, that is, when
  1. solutions consist of few solution components
  2. many variables in the corresponding ILP model have value zero
- LNS will have advantages over CMSA when the opposite is the case

Problem: how to show this?

- Theoretically? hardly possible
- Empirically? Maybe with a parametrizable problem

Example: Multi-dimensional Knapsack Problem (MDKP)

Given:

- A set of items $C = \{1, \ldots, n\}$
- A set of resources $K = \{1, \ldots, m\}$
- Of each resource k we have a maximum quantity c_k (capacity)
- Each item i requires from each resource k a certain quantity $r_{i,k}$
- Each item i has a profit p_i

Valid solutions: Each subset $S \in C$ is a valid solution if

$$\sum_{i \in S} r_{i,k} \leq c_k \quad \forall k \in K$$

Objective function: $f(S) := \sum_{i \in S} p_i$ for all valid solutions S

MDKP: instance tightness

Important parameter: Instance tightness $0 \leq \alpha \leq 1$

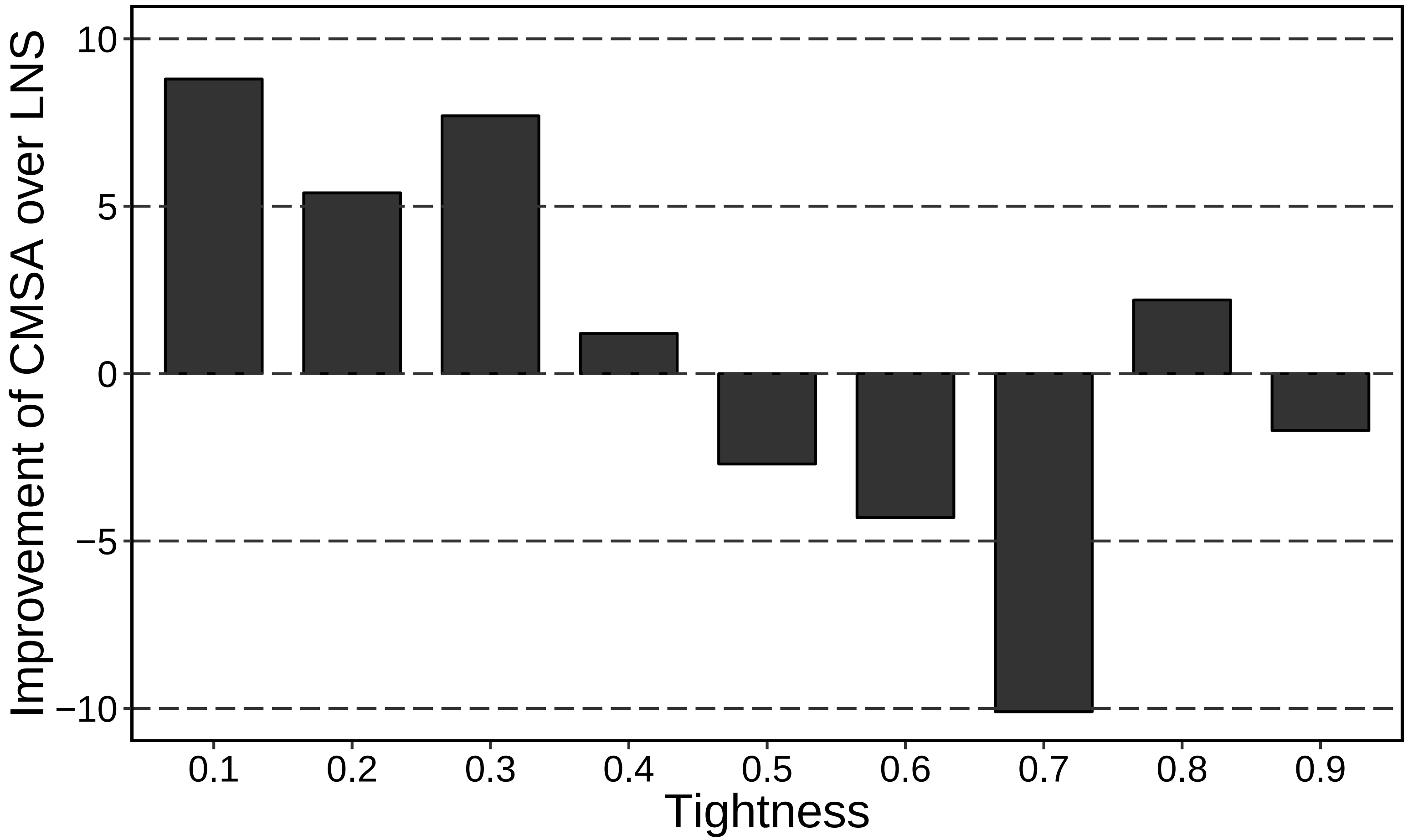
- When α close to zero: capacities are low and valid solution only contain very few items
- When α close to one: capacities are very high and solutions contain nearly all items

Plan:

- Apply both LNS and CMSA to instances from the whole tightness range.
- Both algorithms are tuned with irace seperately for instances of each considered tightness.

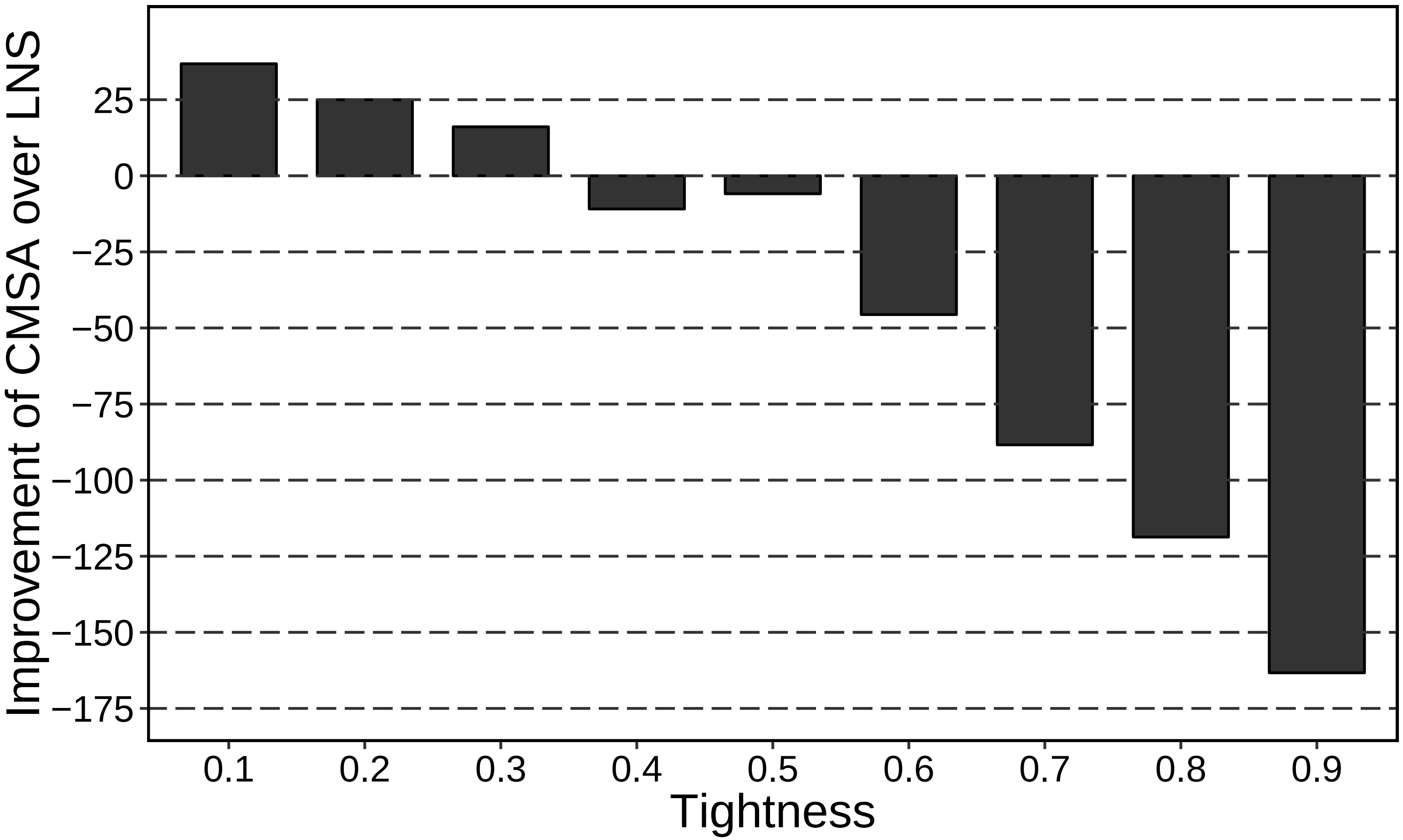
Results for instances with 1000 items

Instance size: $n = 1000$, $m = 10$



Results for instances with 5000 items

Instance size: $n = 5000$, $m = 10$



Summary and Possible Research Directions

Summary:

- Swarm Intelligence: some of our recent/current research topics
- Cmsa: A new hybrid metaheuristic for combinatorial optimization

Possible Research Directions (CMSA):

- Solution construction: adaptive probabilities over time
- A more intelligent version of the aging mechanism
- Taking profit from research on column generation

People involved in certain aspects of this research

Maria J. Blesa

Borja Calvo

Pedro Pinacho

Evelia Lizárraga

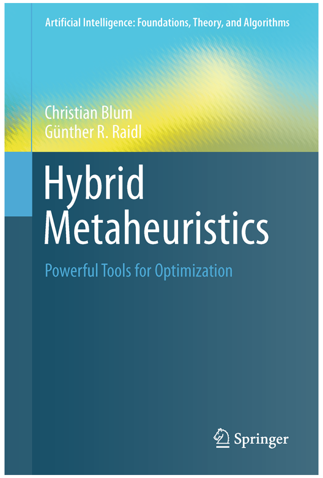
Jóse Antonio Lozano

Manuel López-Ibáñez

Questions?

Literature:

- C. Blum, B. Calvo. **A matheuristic for the minimum weight rooted arborescence problem**. *Journal of Heuristics*, (2015)
- C. Blum, P. Pinacho, J. A. Lozano, M. López-Ibáñez. **Construct, Merge, Solve & Adapt: A new general algorithm for combinatorial optimization**. *Computers & Operations Research*, 2016



New book: C. Blum, G. R. Raidl. Hybrid Metaheuristics – Powerful Tools for Optimization, Springer Series on Artificial Intelligence, 2016