Hybrid Metaheuristics

Christian Blum

Albcom Research Group
Universitat Politècnica De Catalunya
Barcelona, Spain

© www.hemmy.net

Outline of the talk

- **Hybrid Metaheuristics (HMs)**
  - ⋆ Definition
  - ⋆ Classification of HMs
- **Interesting Examples**
  - ⋆ Metaheuristics with Metaheuristics (ILS)
  - ⋆ Metaheuristics with Constraint Programming (ACO)
  - ⋆ Metaheuristics with Tree Search (VNS)
  - ⋆ Metaheuristics with Problem Relaxation (TS)
  - ⋆ Metaheuristics with Dynamic Programming (EC)

Hybrid Metaheuristics

A short introduction

Hybrid metaheuristics (1)

Different metaheuristics: Timeline of introduction

- Simulated Annealing (SA) [Kirkpatrick, 1983]
- Tabu Search (TS) [Glover, 1986]
- Genetic and Evolutionary Computation (EC) [Goldberg, 1989]
- Ant Colony Optimization (ACO) [Dorigo, 1992]
- Greedy Randomized Adaptive Search Procedure (GRASP) [Resende, 1995]
- Particle Swarm Optimiation (PSO) [Eberhart, Kennedy, 1995]
- Guided Local Search (GLS) [Voudouris, 1997]
- Iterated Local Search (ILS) [Stützle, 1999]
- Variable Neighborhood Search (VNS) [Mladenović, 1999]

Hybrid metaheuristics (2)

Definition: What is a hybrid metaheuristic?

- Problem: can not be very well defined

Possible characterization:

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

What is meant by: other techniques for optimiation ?

- Metaheuristics
- Branch & bound
- Dynamic programming
- ILP techniques

Hybrid metaheuristics (3)

Note: Lack of a precise definition is often subject to criticism

History:

- For a long time the different communities co-existed quite isolated
- Hybrid approaches were developed already early, but only sporadically
- Only since about 10 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

Hybrid metaheuristics: classification (1)

References for the classification of hybrid metaheuristics

- C. Cotta. **A study of hybridisation techniques and their application to the design of evolutionary algorithms**, *AI Communications*, 11(3-4):223–224, 1998
- E. Talbi. **A taxonomy of hybrid metheuristics**, *Journal of Heuristics*, 8(5):541–565, 2002
- C. Blum and A. Roli. **Metaheuristics in combinatorial optimization: overview and conceptual comparison**, *ACM Computing Surveys*, 35(3):268–308, 2003
- I. Dumitrescu and T. Stützle. **Combinations of local search and exact algorithms**, In: *Proceedings of Applications of Evolutionary Computation*, volume 2611, Springer LNCS, 2003
- G. Raidl. **A unified view on hybrid metaheuristics**, In: *Proceedings of HM 2006*, volume 4030, Springer LNCS, pages 1–112, 2006

Hybrid metaheuristics: classification (2)

What is hybridized? Metaheuristics with ...

- ... metaheuristics. Examples:
  1. Use of neighborhood-based MHs within population-based MHs
  2. Hyper-heuristics
- ... problem-specific algorithms. Examples:
  1. Continuous optimization: use of gradient-based methods
  2. Simulations for approximating the objective function
- ... other AI/OR techniques. Examples:
  1. Large-scale neighborhood search
  2. Combinations of metaheuristics with constraint programming
- ... a human interactor

Hybrid metaheuristics: classification (3)

What is the level of hybridization?

- High-level: weak coupling
  1. Algorithms retain their own identities
  2. No direct relationship of the internal workings of the algorithms
  3. Interaction over a well-defined interface
- Low-level: strong coupling
  1. Algorithms strongly depend on each other
  2. Individual components or functions are exchanged

Hybrid metaheuristics: classification (4)

What is the control strategy?

- Collaborative
  1. Homogeneous approaches: several instances of the same algorithm
  2. Heterogeneous approaches: for example, A-Teams
- Integrative
  1. Solution merging
  2. Decoder-based approaches
  3. Large-scale neigbhorhood search
  4. Using metaheuristics for finding good upper bounds in branch & bound

Hybrid metaheuristics: classification (5)

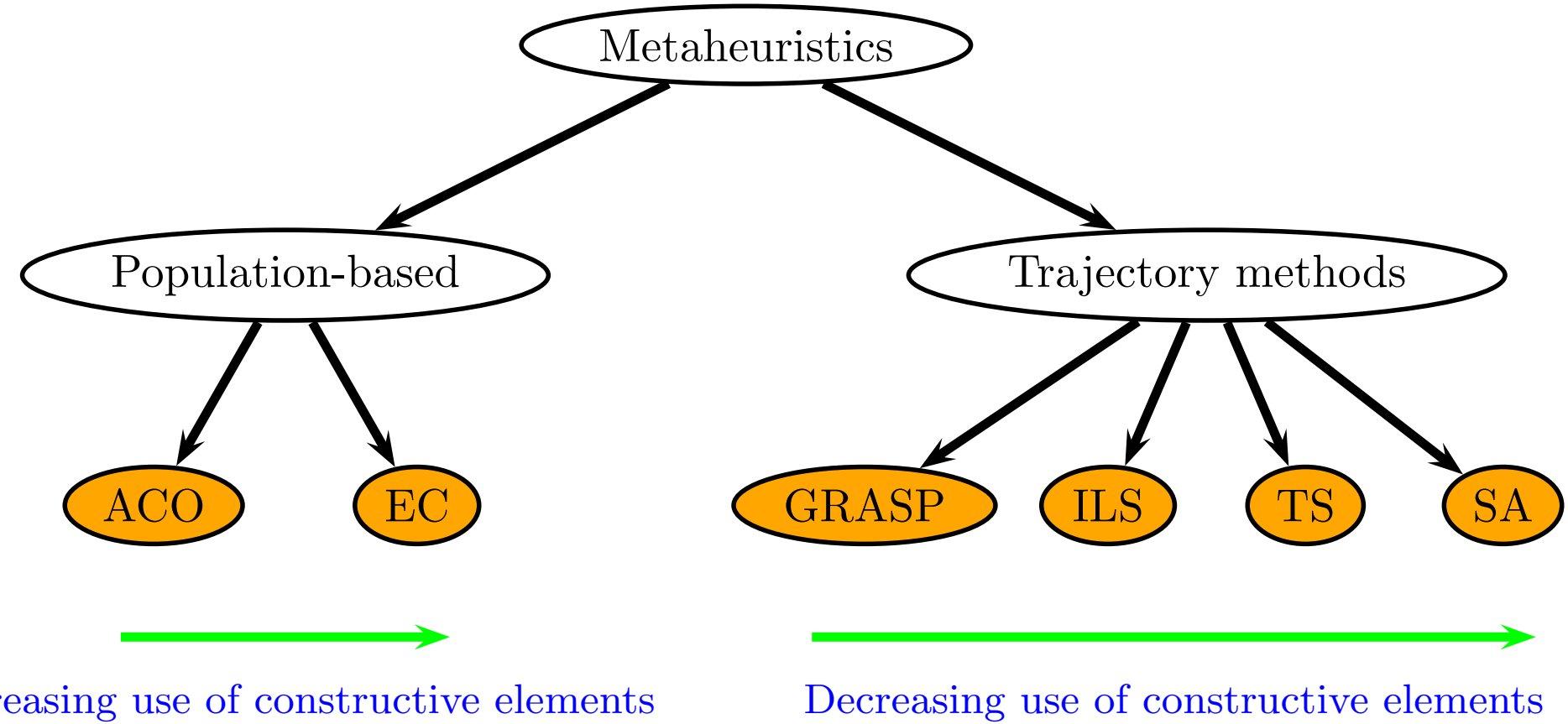
What is the order of execution?

- Sequential: results of earlier executed algorithms are used in later algorithms
- Interleaved. For example:
  1. Using metaheuristics for node selection in branch & bound
  2. Exact algorithms as decoders in decoder-based approaches
- Parallel
  1. Granularity: fine-grained versus coarse-grained
  2. Hardware: homogeneous versus heterogeneous
  3. etc.

Interesting Examples

- **Metaheuristics with Metaheuristics**
- Metaheuristics with Constraint Programming
- Metaheuristics with Tree Search
- Metaheuristics with Problem Relaxation
- Metaheuristics with Dynamic Programming

Characteristics of Different Metaheuristics



- Advantage of pop.-based methods: Diversification ability
- Advantage of trajectory metnods: Intensification ability

What does that mean for Hybridization?

Consequence: Most MH/MH hybrids incorporate trajectory methods into population-based techniques

Examples:

- Application of local search to solutions constructed by ACO
- Application of local search to individuals in evolutionary algorithms (memetic algorithms)

Other examples:

- Population-based iterated local search
- Multi-level techniques

Iterated Local Search

Pseudo-code:

1: $s \leftarrow$ GenerateInitialSolution()
2: $s \leftarrow$ LocalSearch(s)
3: **while** termination conditions not met **do**
4: $\quad s' \leftarrow$ Perturbation($s, history$)
5: $\quad \hat{s}' \leftarrow$ LocalSearch(s')
6: $\quad s \leftarrow$ ApplyAcceptanceCriterion($\hat{s}', s, history$)
7: **end while**

Key components:

- Perturbation mechanism
- Local search

Population-Based Iterated Local Search

1: $P \leftarrow$ GenerateInitialPopulation(n)
2: Apply LocalSearch() to all $s \in P$
3: **while** termination conditions not met **do**
4: $P' \leftarrow P$
5: **for** all $s \in P$ **do**
6: $s' \leftarrow$ Perturbation($s, history$)
7: $\hat{s}' \leftarrow$ LocalSearch(s')
8: $P' \leftarrow P' \cup \{\hat{s}'\}$
9: **end for**
10: $P \leftarrow$ Best n solution from P'
11: **end while**

Main reference: T. Stützle. **Iterated local search for the quadratic assignment problem**, *European Journal of Operational Research*, 174(3):1529–1539, 2006

The Multi-level Framework (1)

General references:

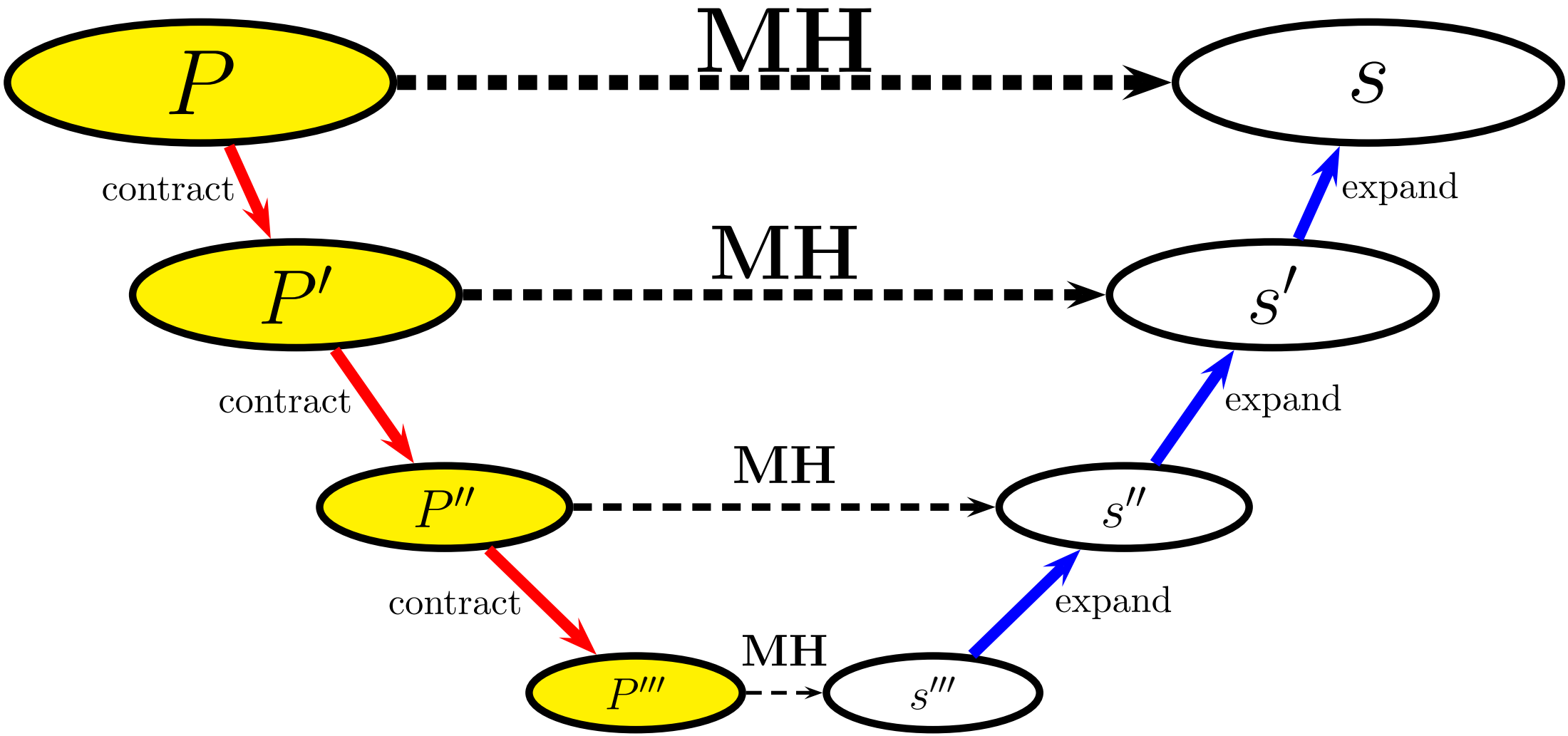
- C. Walshaw. **Multilevel refinement for combinatorial optimisation**, *Annals of Operations Research*, 131:325–372, 2004
- C. Walshaw. **Multilevel refinement for combinatorial optimisation: boosting metaheuristic performance**, In: *Hybrid Metaheuristics–An Emerging Approach to Optimization*, volume 114 of Studies in Computational Intelligence, pages 261–289, Springer Verlag, Berlin, Germany, 2008

General idea:

- **First:** Iterative coarsening of the original problem instance
- **Then:** Find a solution to the coarsest level
- **Finally:** Iteratively refine this solution at each level

The Multi-level Framework (2)

The multi-level framework:



The Multi-level Framework (3)

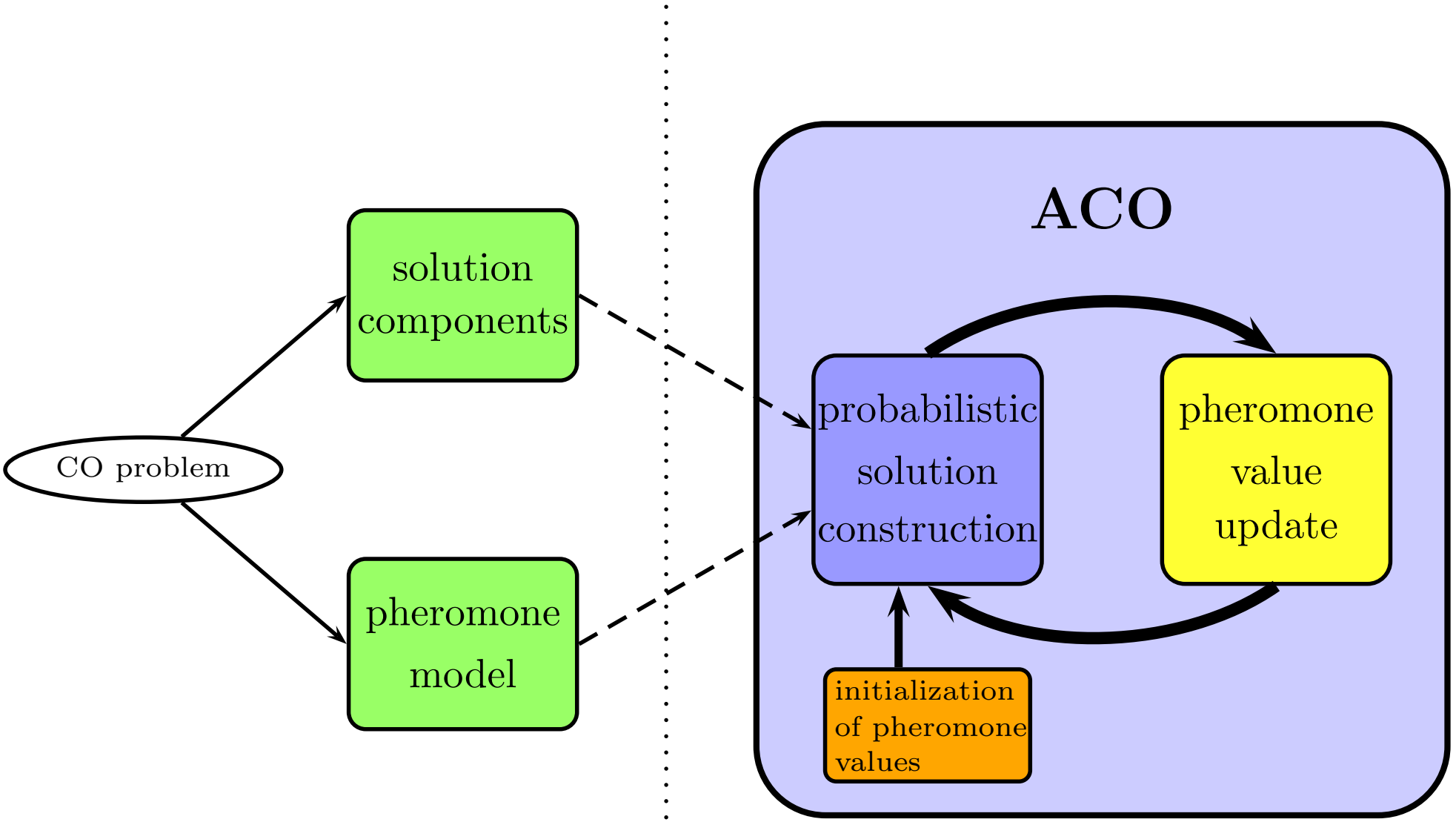
Specific references:

- P. Korosec, J. Silc, and B. Robic. **Solving the mesh-partitioning problem with an ant-colony algorithm**, *Parallel Computing*, 30(5-6):785-801, 2004
- C. Walshaw. **A multilevel approach to the travelling salesman problem**, *Operations Research*, 50(5):862–877, 2002
- T. G. Crainic, Y. Li, and M. Toulouse. **A first multilevel cooperative algorithm for capacitated multicommodity network design**, *Computers & Operations Research*, 33(9):2602–2622, 2006

Interesting Examples

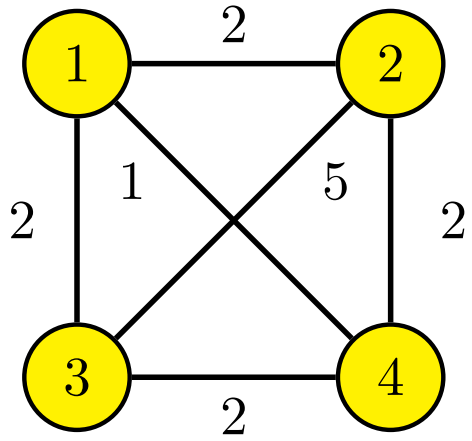
- Metaheuristics with Metaheuristics
- **Metaheuristics with Constraint Programming**
- Metaheuristics with Tree Search
- Metaheuristics with Problem Relaxation
- Metaheuristics with Dynamic Programming

Ant Colony Optimization (ACO)



Traveling salesman problem (TSP)

Given: A completely connected, undirected graph $G = (V, E)$ with edge-weights.

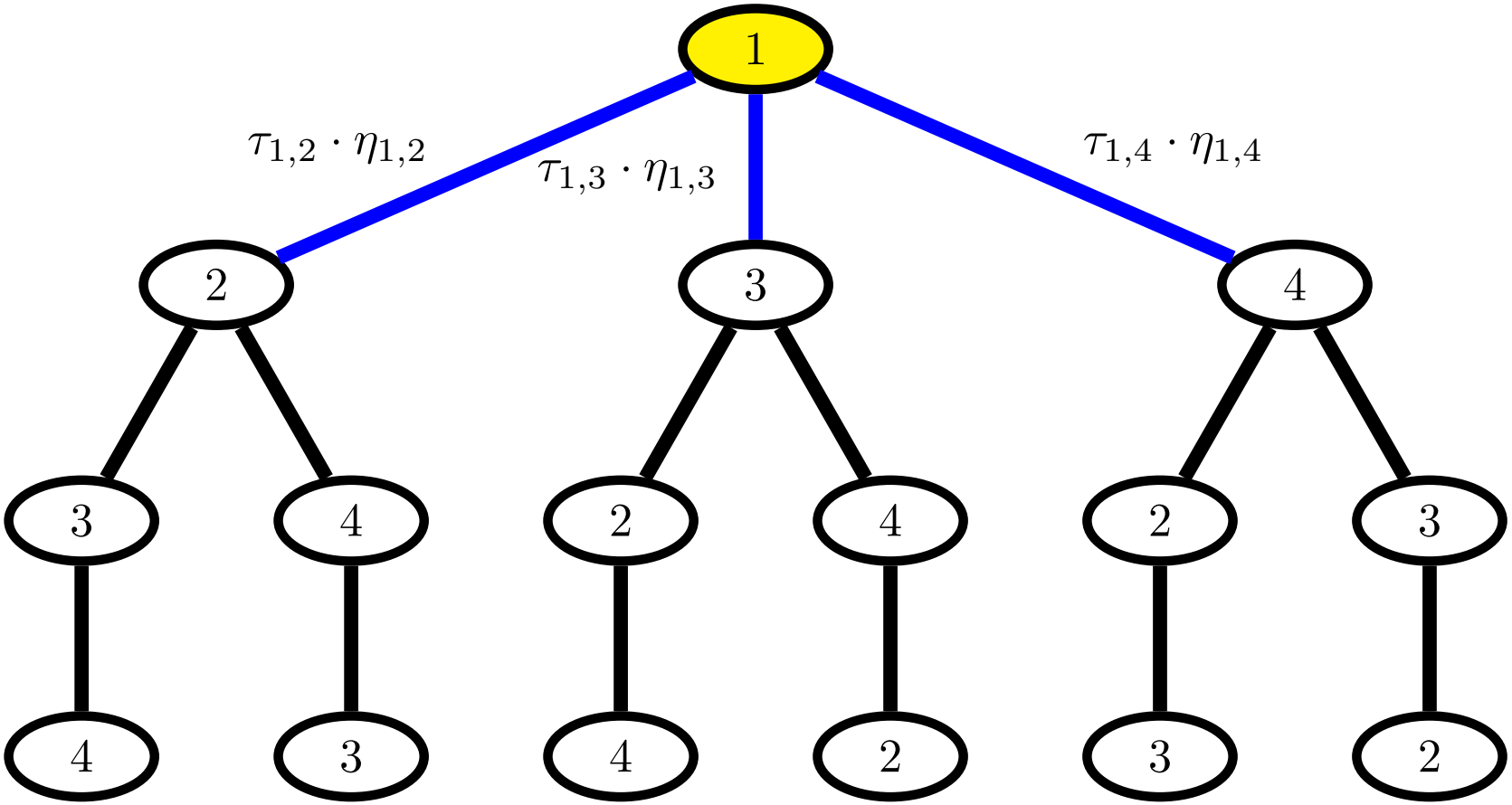


Goal: Find a tour (a Hamiltonian cycle) in G with minimal sum of edge weights.

Solution construction: Constructive mechanism of the nearest-neighbor heuristic (starting from city 1)

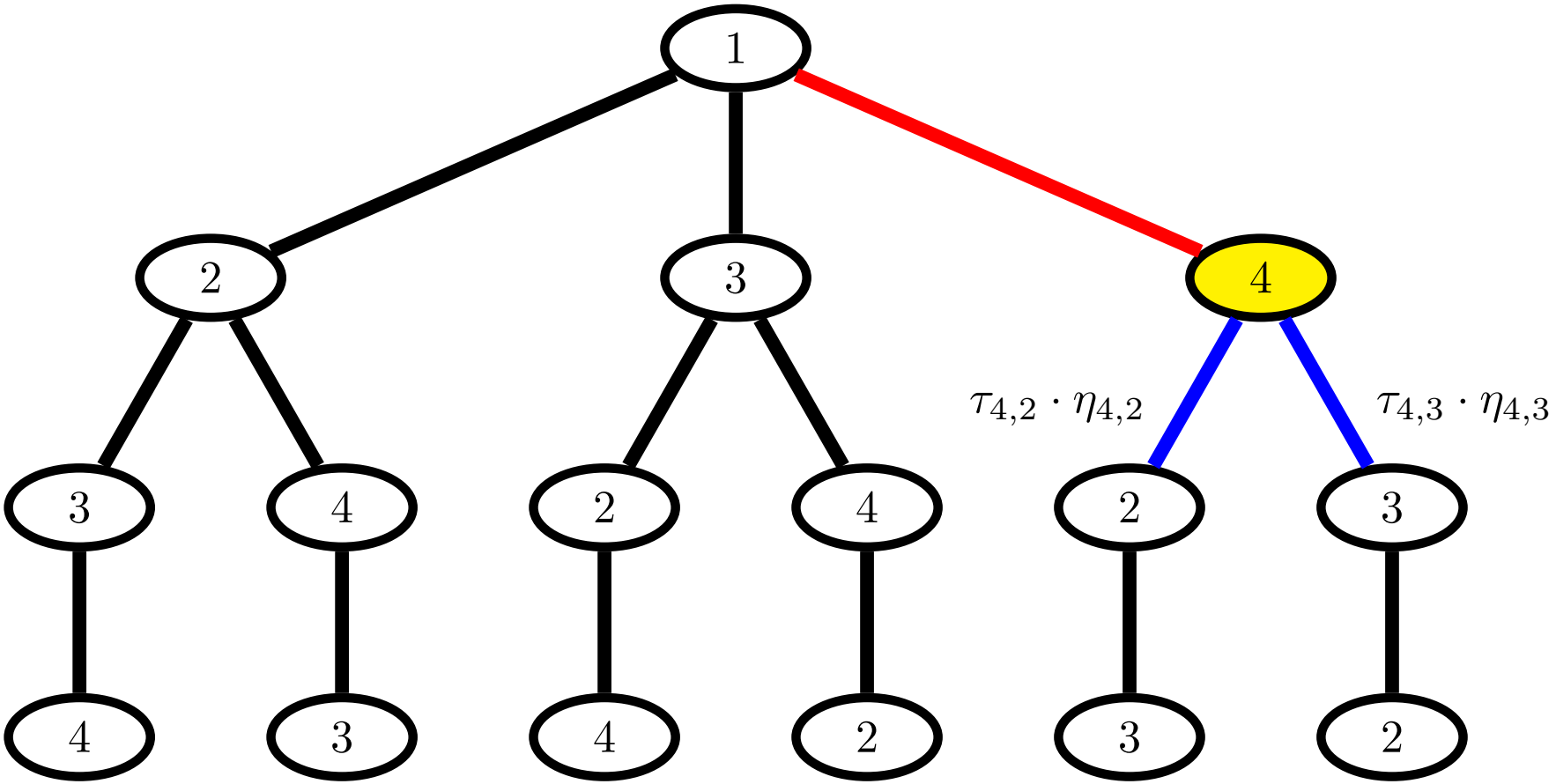
ACO as a tree search algorithm

1st construction step:



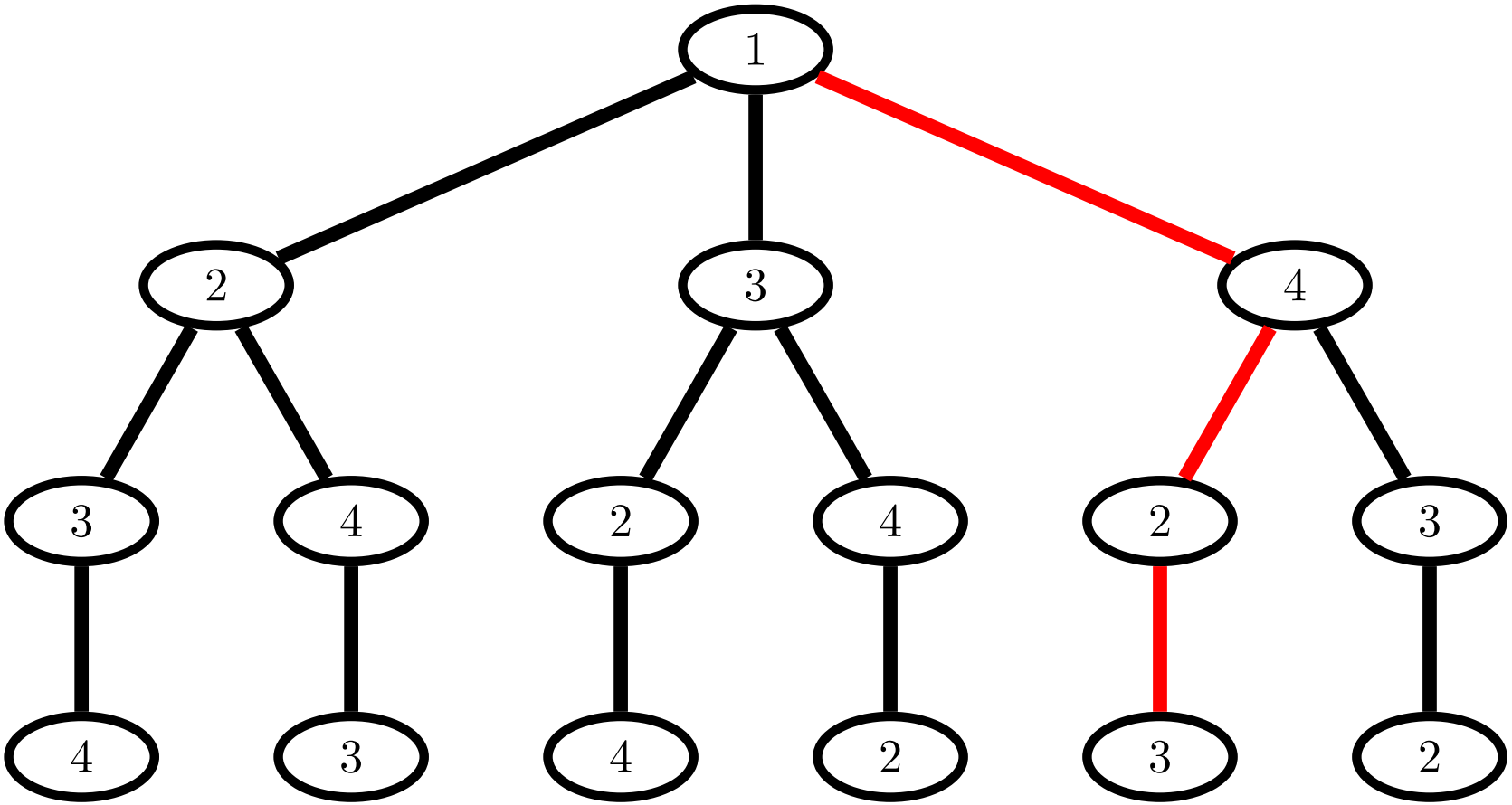
ACO as a tree search algorithm

2nd construction step:



ACO as a tree search algorithm

3rd construction step:



ACO hybridized with constraint programming (1)

References:

- B. Meyer and A. Ernst. **Integrating ACO and Constraint Propagation**, In: *Proceedings of ANTS 2004*, volume 3172 of Springer LNCS, pages 166–177, 2004
- M. Khichane, P. Albert, and C. Solnon. **CP with ACO**, In: *Proceedings of CPAIOR 2008*, volume 5015 of Springer LNCS, pages 328–332, 2008

General idea:

- Successively reduce the variable domains by contraint propagation
- Let ACO search the reduced search tree

ACO hybridized with constraint programming (2)

Constraint programming (CP): Computational systems based on constraints

How does it work?

- Phase 1:
  - ⋆ Express CO problem in terms of a discrete problem (variables+domains)
  - ⋆ Define ("post") constraints among the variables
  - ⋆ The constraint solver reduces the variable domains
    - ... before solution construction
    - ... during solution construction
- Phase 2: Labelling
  - ⋆ Search through the remaining search tree
  - ⋆ Possibly "post" additional constraints

ACO hybridized with constraint programming (3)

Simple example: **minimize** $f(X, Y, Z) \mapsto \mathbf{R}$

subject to

$$X \in \{1, \ldots, 8\}$$
$$Y, Z \in \{1, \ldots, 10\}$$
$$X \neq 7,\ Z \neq 2$$
$$X - Z = 3Y$$

Constraint propagation:

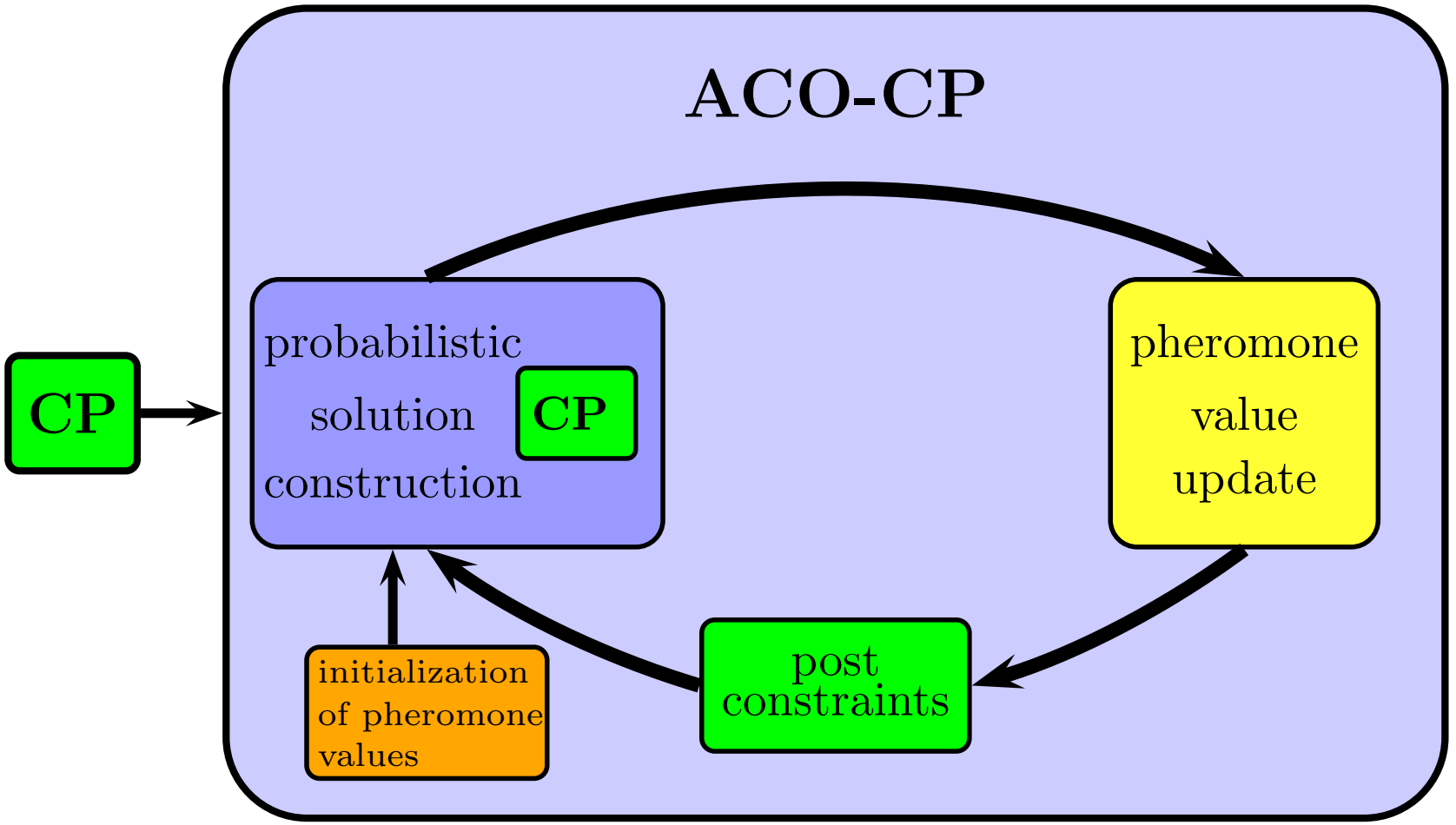
- Step 1: Use $X \neq 7$ and $Z \neq 2$
  1. $X \in \{1, \ldots, 6, 8\}$
  2. $Z \in \{1, 3, \ldots, 10\}$

ACO hybridized with constraint programming (4)

- Step 2: Use $X - Z = 3Y$
  1. Because of the domains of X and Z: $X - Z < 8$
  2. $\Rightarrow$ $3Y < 8$
  3. $\Rightarrow$ $Y \leq 2$
  4. $\Rightarrow$ $Y \in \{1, 2\}$
- Step 3: Use again $X - Z = 3Y$
  1. Because of the reduced domain of Y: $3Y \geq 3$
  2. $\Rightarrow$ $X - Z \geq 3$
  3. $\Rightarrow$ $X \in \{4, 5, 6, 8\}$ and $Z \in \{1, 3, 4, 5\}$

ACO hybridized with constraint programming (5)

ACO-CP hybrid:



ACO hybridized with constraint programming (6)

Evaluation:

- Advantage of ACO:
  Good in finding high quality solutions for moderately constrained problems
- Advantage of CP:
  Good in finding feasible solutions for highly constrained problems

ACO-CP: Good with intermediate number of feasible solutions

Problem:

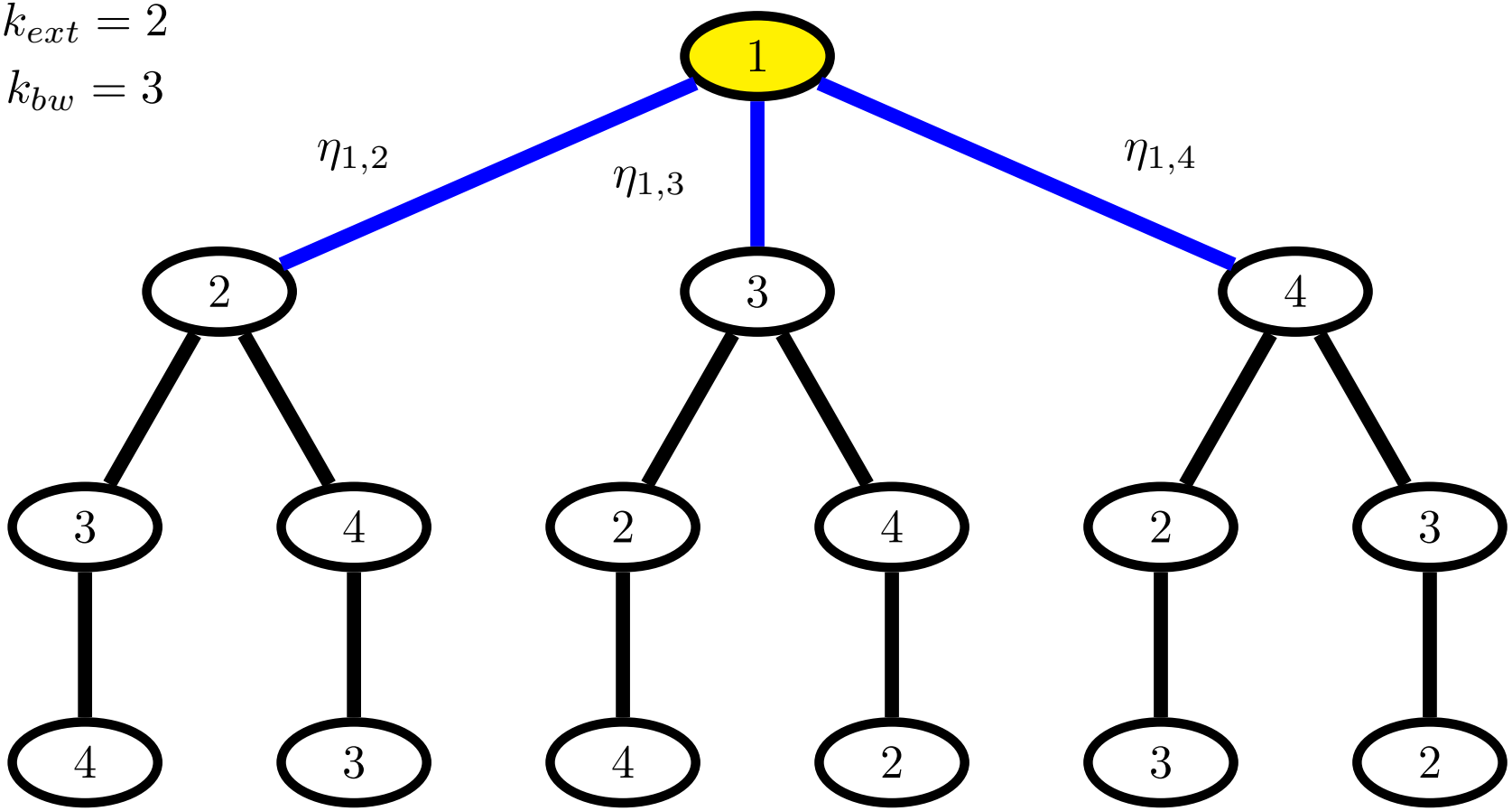
- Constraint propagation takes a lot of time
- Moreover: contraint propagation is repeated many times

Beam search

1st construction step:

$k_{ext} = 2$

$k_{bw} = 3$

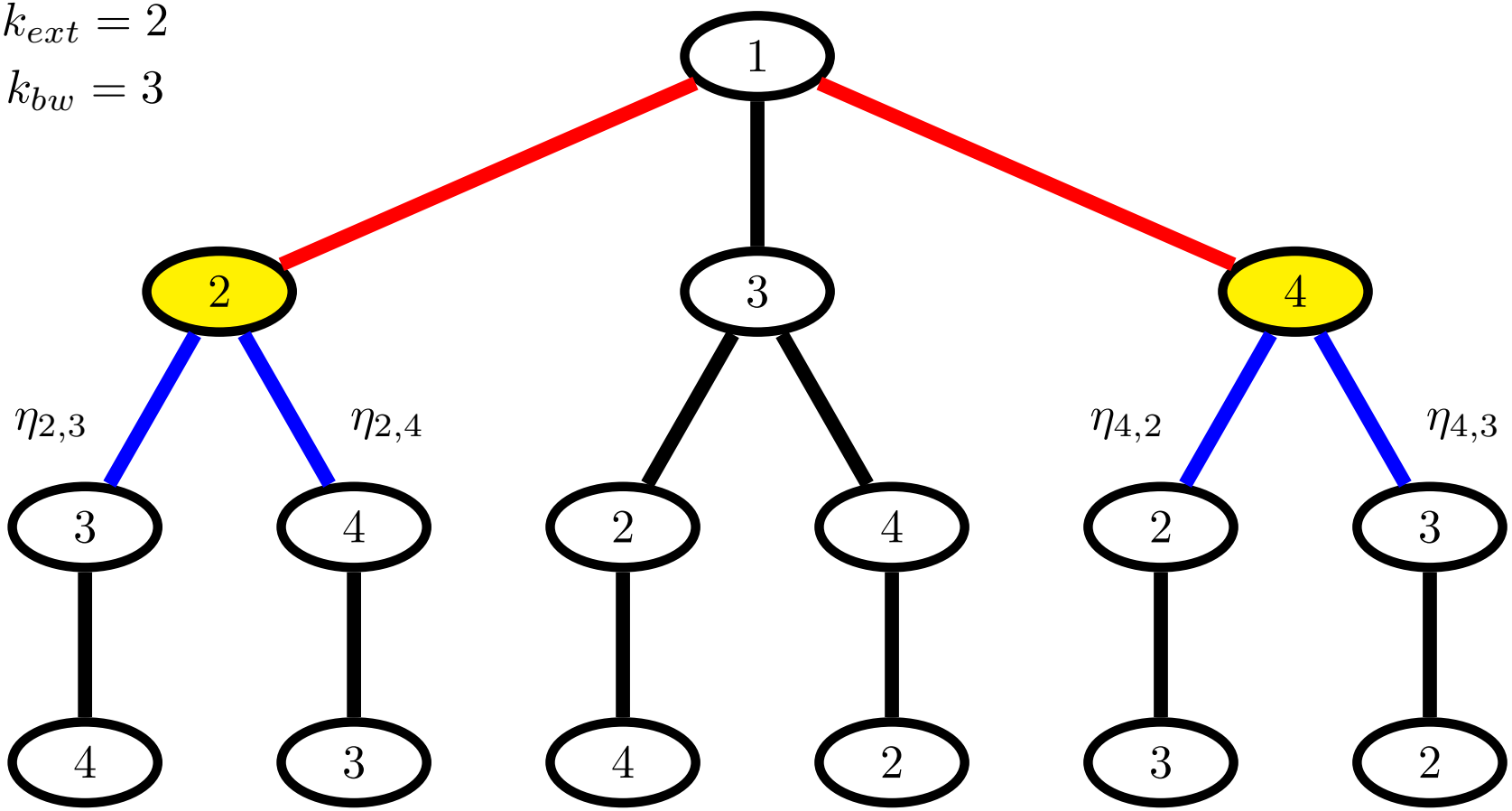


Beam search

2nd construction step:

$k_{ext} = 2$

$k_{bw} = 3$

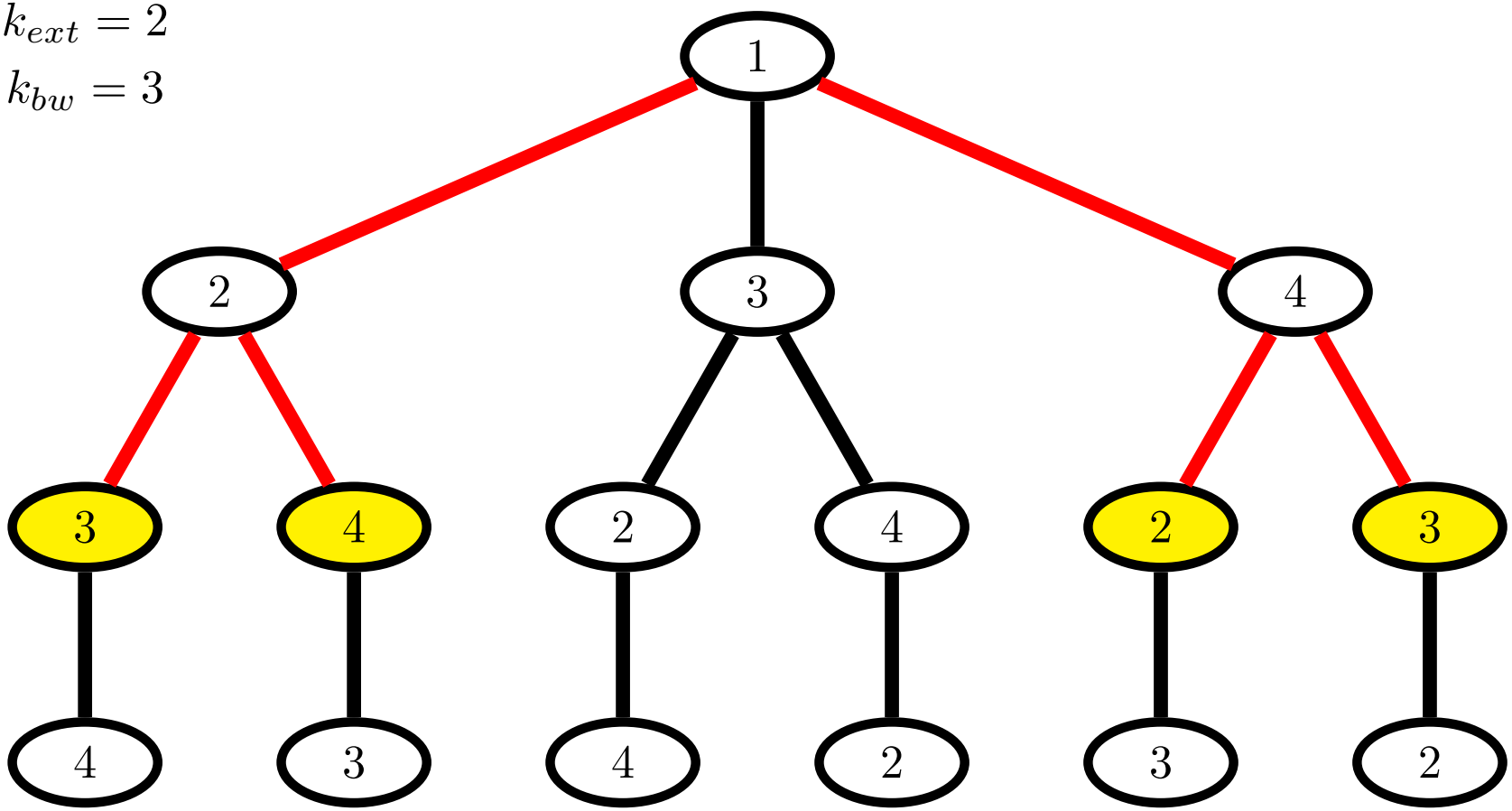


Beam search

After 2nd construction step: use lower bound

$k_{ext} = 2$

$k_{bw} = 3$

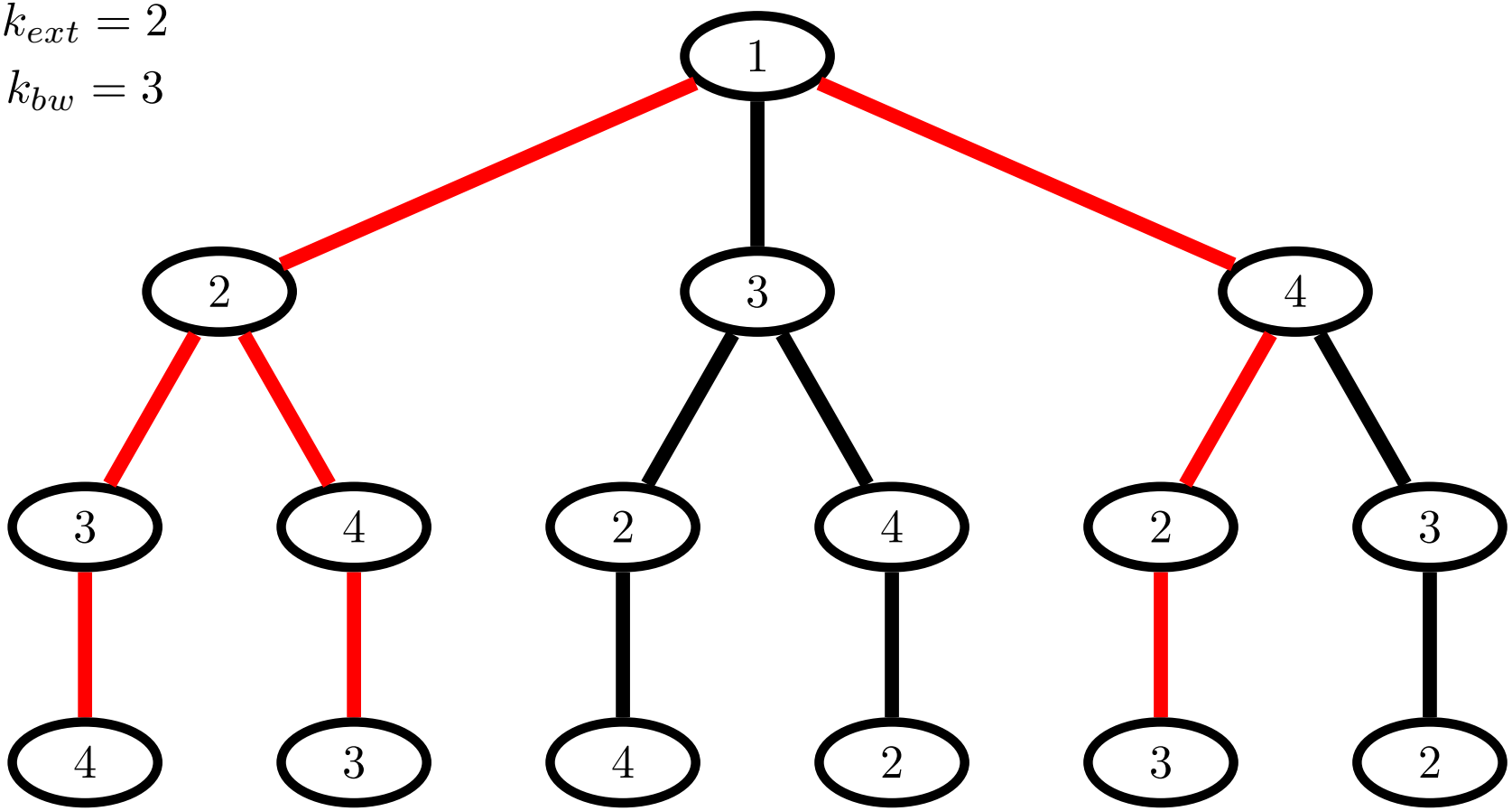


Beam search

3rd construction step:

$k_{ext} = 2$

$k_{bw} = 3$



Hybrid algorithm: Beam-ACO

Idea:

- Instead of n_a independent solution constructions per iteration,
- perform a probabilistic beam search with beam width $k_{bw} = n_a$

Advantages:

- Strong heuristic guidance by a lower bound
- Embedded in the adaptive framework of ACO

Hybrid algorithm: Beam-ACO

Applications Beam-ACO was applied to the following problems:

- **Open shop scheduling (OSS)**
  Blum, *Computers & Operations Research* (2005)
- **Longest common subsequence (LCS) problem**
  Blum, Mastrolilli, *HM 2007*
- **Supply chain management**
  Caldeira et al., *FUZZ-IEEE 2007*, *ISFA 2007*
- **Simple assembly line balancing (SALB)**
  Blum, *INFORMS Journal on Computing* (2008)
- **Travelling salesman problem with time windows (TSPTW)**
  López-Ibañez et al., *EvoCOP 2009*

Hybrid algorithm: Beam-ACO

Question: Why does it work so well?

Observation: Beam-ACO uses 2 types of complementary problem information

1. A greedy function
2. Lower (respectively, upper) information

These two types of information are especially well exploited in Beam-ACO!

Interesting Examples

- Metaheuristics with Metaheuristics
- Metaheuristics with Constraint Programming
- **Metaheuristics with Tree Search**
- Metaheuristics with Problem Relaxation
- Metaheuristics with Dynamic Programming

Large-scale neighborhood search (1)

General references:

- R. K. Ahuja, O. Ergun, J. B. Orlin, and A. P. Punnen. **A survey of very large-scale neighborhood search techniques**, *Discrete Applied Mathematics*, 123(1-3):75–102, 2002
- M. Chiarandini, I. Dumitrescu, and T. Stützle. **Very Large-Scale Neighborhood Search: Overview and Case Studies on Coloring Problems**, In: *Hybrid Metaheuristics–An Emerging Approach to Optimization*, volume 114 of Studies in Computational Intelligence, pages 117–150, Springer Verlag, Berlin, Germany, 2008

Key issues in local search:

- Defining an appropriate neighborhood structure
- Choosing a way of examining the neighborhood of a solution

Large-scale neighborhood search (2)

General tradeoff:

- Small neighborhoods:
  1. **Advantage:** It is fast to find an improving neighbor (if any)
  2. **Disadvantag:** The average quality of the local minima is low
- Large-scale 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
- In some cases an efficient exact technique may exist

Example: Biological Background

- **Given:** A set of haplotype sequences from a population of individuals
- **Goal:** Study the evolutionary history of the chosen individuals
- **Important for** the discovery of the genetic basis of complex diseases

In case the population has evolved from a relatively small set of founders, the evolutionary history can be studied by trying to reconstruct the haplotype sequences from founder fragments

- Problem: Generally, neither the founder sequences nor their number are known

The Founder Sequence Reconstruction Problem (FSRP)

- Given: A set of m recombinants $\mathcal{C} = \{C_1, \ldots, C_m\}$
  - ⋆ **Here:** $\forall i, C_i$ is a binary string of length n
- Candidate solution: A set of k founders $\mathcal{F} = \{F_1, \ldots, F_k\}$
  - ⋆ **Here:** $\forall j, F_j$ is a binary string of length n
- A solution is valid if $\mathcal{C}$ can be reconstructed from $\mathcal{F}$.
- This is the case when each $C_i \in \mathcal{C}$ can be decomposed into a sequence of $p_i \leq n$ fragments $Fr_{i1}Fr_{i2}\ldots Fr_{ip_i}$, such that each fragment Fr_{ij} appears at the same position in at least one of the founders
- Given $\mathcal{F}$, a minimal decomposition, where the number of breakpoints $\sum_{i=1}^{n} p_i - n$ is minimal, can be derived in polynomial time

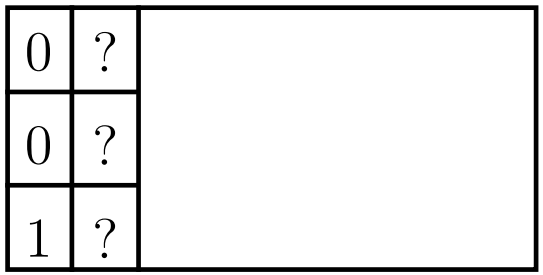
FSRP: Optimization Goal, and Example

- Optimization goal: Given k, find a valid solution $\mathcal{F}^*$ that minimizes $f(\cdot)$

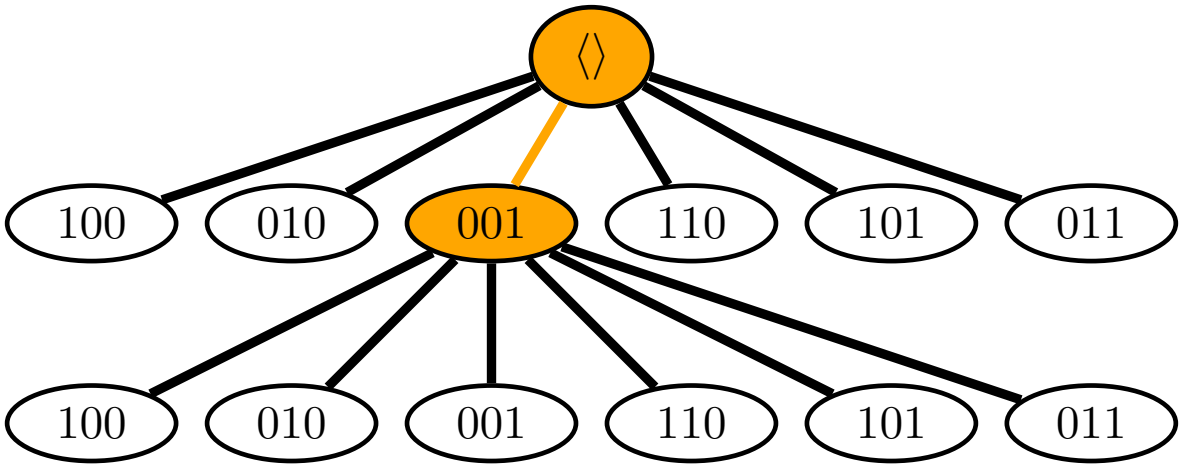
Example:

Recombinants $\mathcal{C}$	Founders $\mathcal{F}$	Reconstruction
1 1 0 1 1 0 1		b b b b b|c c
1 0 1 0 0 0 1	0 1 1 0 1 0 0	c c c c c c c
0 1 1 1 1 1 1	1 1 0 1 1 1 1	a a a|b b b b
0 1 1 0 1 0 0	1 0 1 0 0 0 1	a a a a a a a
1 1 0 0 0 1 1		b b b|c c|b b

Branch & Bound Algorithm: RECBLOCK



Partial solution



Search Tree

Wu, Y., Gusfield, D. **Improved algorithms for inferring the minimum mosaic of a set of recombinants**. In: *Proceedings of CPM 2007*, Volume 4580 of LNCS, Springer Verlag, Berlin (2007), pages 150–161

Branch & Bound Algorithm: RECBLOCK

Observation: Given some fixed founders, RECBLOCK can be used to obtain the optimal setting for the remaining founders

Example: 4 fixed founders $\{1, 2, 4, 7\}$, and 3 missing founders $\{3, 5, 6\}$

Variable Neighborhood Descent (VND)

Observation: VND is a heuristic version of variable neighborhood search (VNS)

1: INPUT: a solution s, $r_{\max}$ neighborhood functions
2: $r := 1$
3: **while** $r \leq r_{\max}$ **do**
4: $s' :=$ PickBestNeighbor$(s, \mathcal{N}_r)$
5: **if** $f(s') < f(s)$ **then**
6: $s := s'$
7: $r := 1$
8: **else**
9: $r := r + 1$
10: **end if**
11: **end while**
12: OUTPUT: a (possibly) improved solution s

Hybrid VND for the FSRP

1: INPUT: a solution s, number k of founders
2: $r := 1$
3: **while** $r \leq k$ **do**
4: $\hat{s} :=$ DeleteFounders(s, r)
5: $s' :=$ RECBLOCK$(\hat{s})$
6: **if** $f(s') < f(s)$ **then**
7: $s := s'$
8: $r := 1$
9: **else**
10: **if** maximal number of trials reached **then** $r := r + 1$
11: **end if**
12: **end while**
13: OUTPUT: a (possibly) improved solution s

Interesting Examples

- Metaheuristics with Metaheuristics
- Metaheuristics with Constraint Programming
- Metaheuristics with Tree Search
- **Metaheuristics with Problem Relaxation**
- Metaheuristics with Dynamic Programming

Problem Relaxation

Observe: Problem relaxations can be obtained (among others) by

- Simplifying constraints of an IP formulation
- Dropping constraints of an IP formulation
  (e.g. integrality contraints)
- Moving constraints in terms of penalties to the objective function
  (e.g. Lagrangian relaxation)

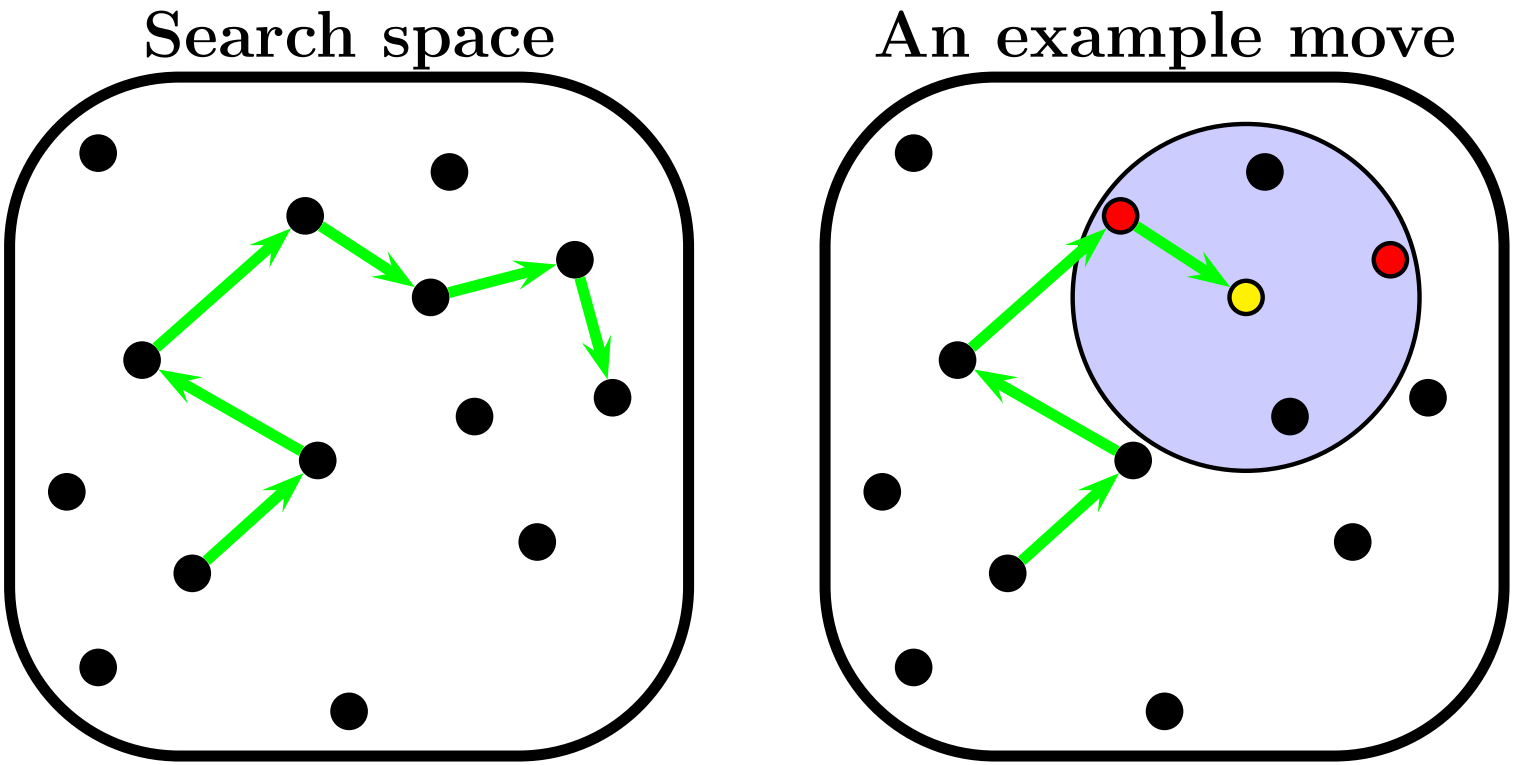
Use of relaxations:

- As bounds for branch & bound algorithms
- As approximation for integer solutions
- As heuristic information for solution construction

Tabu Search

Main feature: Use of tabu lists for storing solution features

Note: Tabu lists are used to avoid going back to already visited solutions



Hybrid Tabu Search

Specific Reference:

- M. Vasquez and Y. Vimont. **Improved results on the 0–1 multidimensional knapsack problem.** *European Journal of Operational Research*, 165(1):70–81, 2005

Characteristics:

- Collaborative hybridization approach
- 1st algorithm phase: problem relaxation is used to produce a bunch of promising solutions
- 2nd algorithm phase: tabu search is used to search around these solutions

The 0-1 Multidimensional Knapsack Problem (MKP)

Given:

- n objects, each object i with a profit c_i
- m resources, each resource j with a capacity b_j
- Each object i has a requirement a_{ij} of each resource j

IP formulation:

$$\textbf{max} \sum_{i=1}^{n} c_i \cdot x_i$$

subject to

$$a_{ij} \cdot x_i \leq b_j \qquad j = 1, \ldots, m$$
$$x_i \in \{0, 1\} \qquad i = 1, \ldots, n$$

First Algorithm Phase

Main ideas:

- Dropping the integrality contraints
- For all k such that $0 \leq k_{\min} \leq k \leq k_{\max} \leq n$ solve

$$\textbf{max} \sum_{i=1}^{n} c_i \cdot x_i$$

subject to

$$a_{ij} \cdot x_i \leq b_j \qquad j = 1, \ldots, m$$
$$0 \leq x_i \leq 1 \qquad i = 1, \ldots, n$$
$$\sum_{i=1}^{n} x_i = k$$

Second Algorithm Phase

Main ideas:

- From the 1st phase solutions: Produce integer solutions by rounding
- Use tabu search to search in the vicinity of these integer solutions
- Definition of vicinity: maximum Hamming distance

Interesting Examples

- Metaheuristics with Metaheuristics
- Metaheuristics with Constraint Programming
- Metaheuristics with Tree Search
- Metaheuristics with Problem Relaxation
- **Metaheuristics with Dynamic Programming**

Dynamic Programming

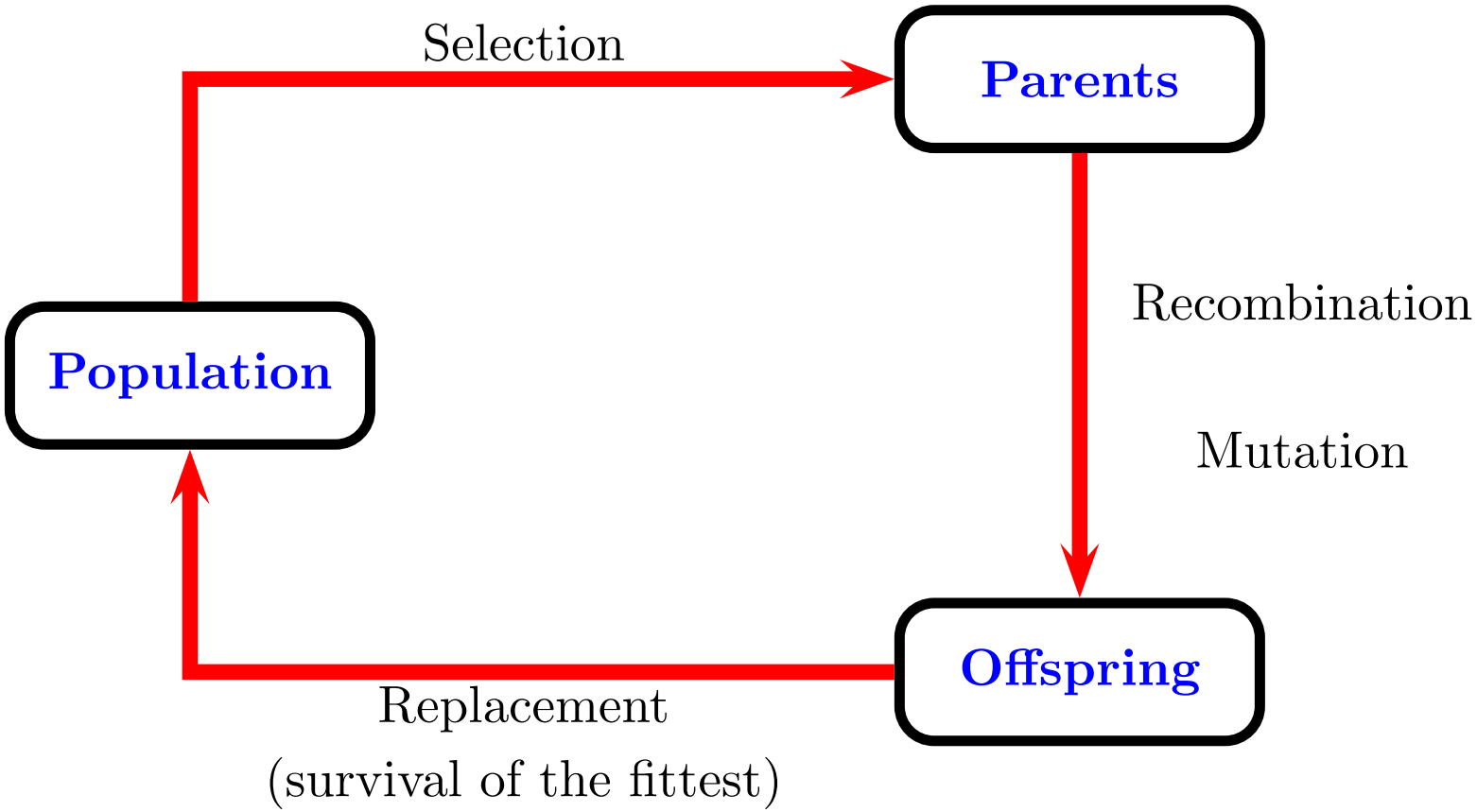
How does it work?

1. Divide the given problem into sub-problems
2. Combine solutions of already solved sub-problems to solutions to bigger sub-problems until a solution for the original problem is obtained

Required properties of the problem

1. Optimal substructure: Optimal solution to the problem must contain optimal solutions to sub-problems
2. Space of sub-problems: Should be of moderate size (polynomial)

Evolutionary Algorithms (EAs)



The k-Cardinality Tree (KCT) Problem (1)

Specific reference:

- C. Blum. **A new hybrid evolutionary algorithm for the k-cardinality tree problem**, In: *Proceedings of GECCO 2006*, ACM Press, pages 515–522, 2006

Definition: The k-cardinality tree problem

Given:

- An undirected graph $G = (V, E)$,
- Edge-weights w_e, $\forall\, e \in E$, and node-weights w_v, $\forall\, v \in V$.
- A cardinality $k < |V|$

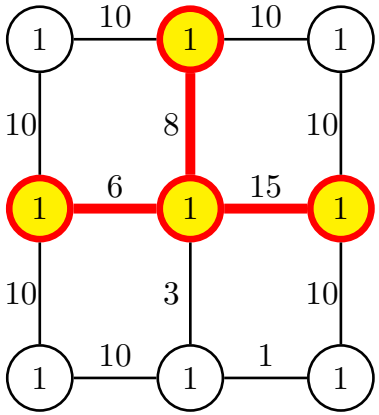
The k-Cardinality Tree (KCT) Problem (2)

Let $\mathcal{T}_k$ be the set of all trees in G with exactly k edges

Optimization goal: Find a k-cardinality tree $T_k \in \mathcal{T}_k$ which minimizes

$$f(T_k) = \left(\sum_{e \in E(T_k)} w_e \right) + \left(\sum_{v \in V(T_k)} w_v \right)$$

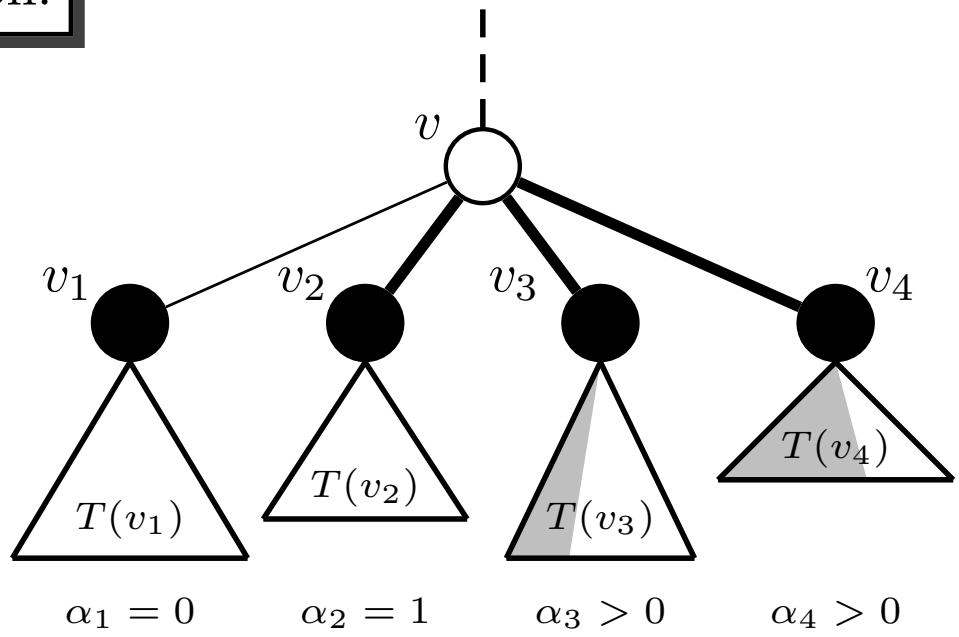
Example: A 3-cardinality tree



Dynamic Programming for the KCT Problem

Observation: KCT can be solved optimaly if G is a tree

Graphical explanation:



Complexity: $\mathcal{O}(k^2|V|)$

Utilizing DP for Crossover

Given: Two k-trees T_1 and T_2 (parents)

Case 1: T_1 and T_2 have a least one node in common

1. Merge T_1 and $T_2 \leftrightarrow$ A graph G_c
2. Generage a minimum spanning tree T of G_c
3. Use DP for obtaining the best k-tree in T

Case 2: T_1 and T_2 do not have any node in common

1. Use tree construction to increase T_1 until it touches $T_2 \leftrightarrow T$
2. Use DP for obtaining the best k-tree in T

Summary and Conclusions

Presented topics:

- Hybrid metaheuristics: a short intro
- Despite criticism: term *hybrid metaheuristics* is useful
- Five representative hybridization examples

Bottom line: More and more state-of-the-art methods are hybrids

But: Still a lot of space for new, conceptually different hybrids!!

(Potentially) Useful Books