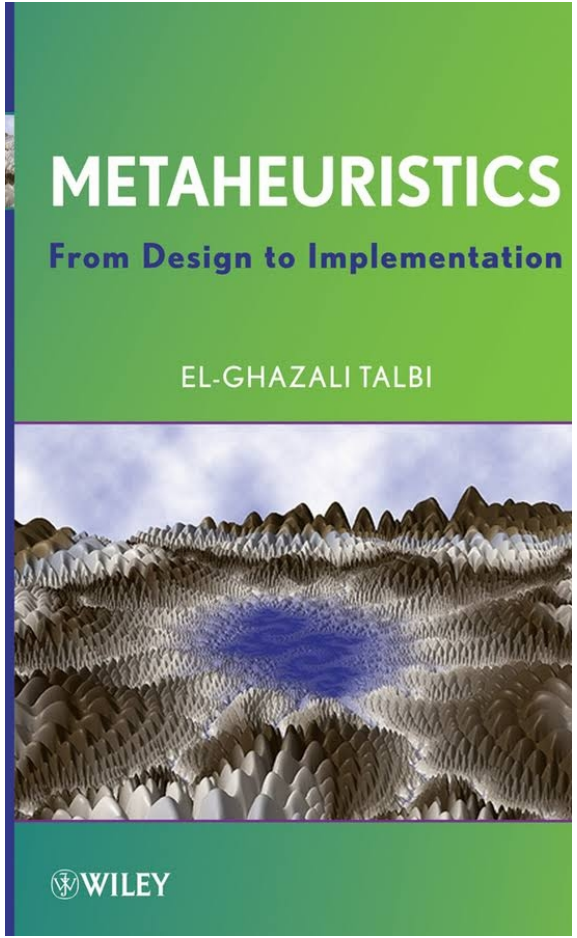


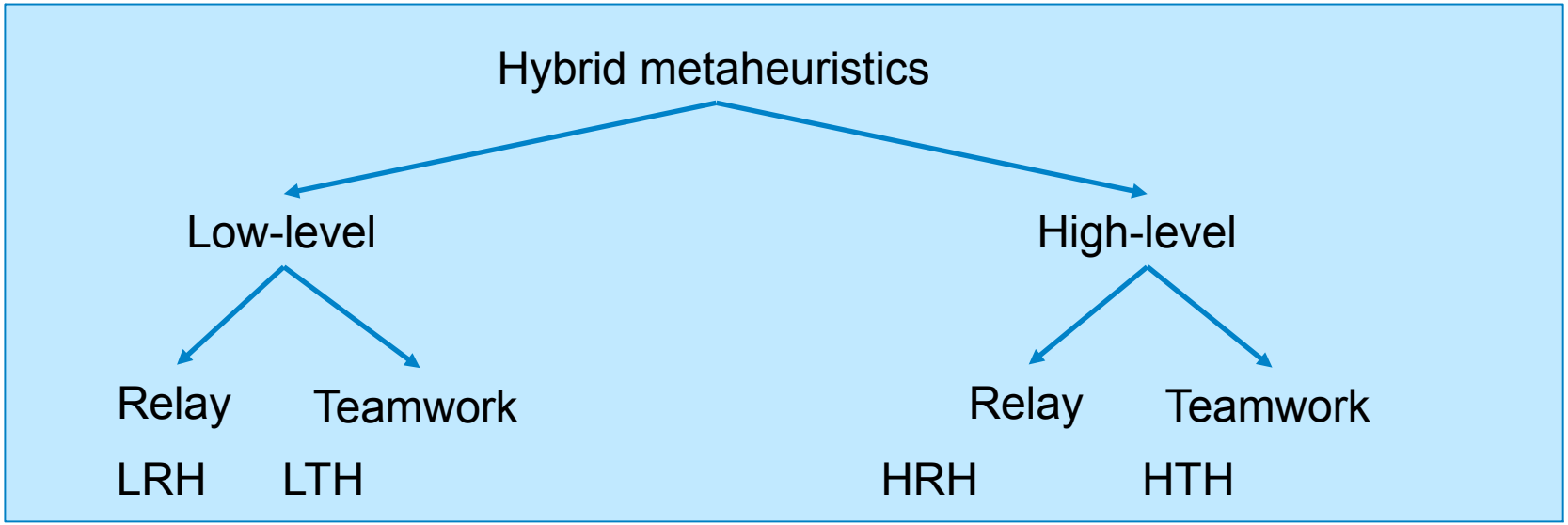
Chapter 5.2 and 5.3

Sara Gestrelius

PART OF RI.SE

SWEDISH ICT SICS

Last time: Classification



SWEDISH ICT SICS

5.2 Combining metaheuristics with mathematical programming

Mathematical Programming
Low-level replay
Low-level teamwork
High-level relay
High-level teamwork

Mathematical Programming

MP vs MH

Enumerative algorithms

Relaxation and decomposition methods

Cutting Plane and Pricing algorithms

Why?

Mathematical Programming: Exact but expensive in memory and time.

Metaheuristics: Not exact but manageable in memory and time.

(mainly if your problem is discrete and/or have nonlinear hard constraints/objective function)

Enumerative algorithms

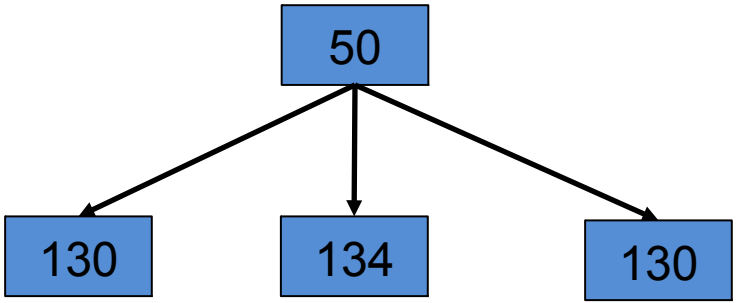
Branch and bound:

1. Branch: Divide problem e.g. x=1 x=0 if x binary.
2. Bound: Keep track of upper bound (i.e. best solution found so far) and also lower bound for all nodes.
3. Prune: If lower bound > upper bound, don't investigate that node further.

Mathematical Programming

Enumerative algorithms

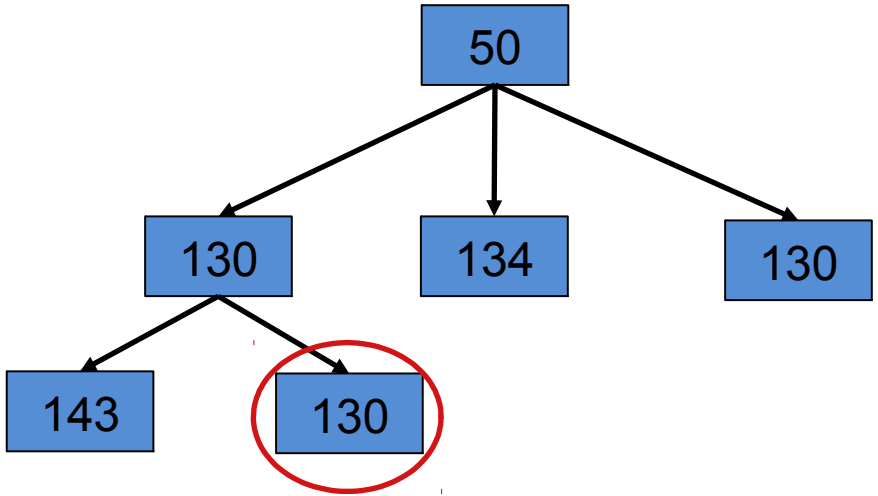
Branch and bound:



Mathematical Programming

Enumerative algorithms

Branch and bound:

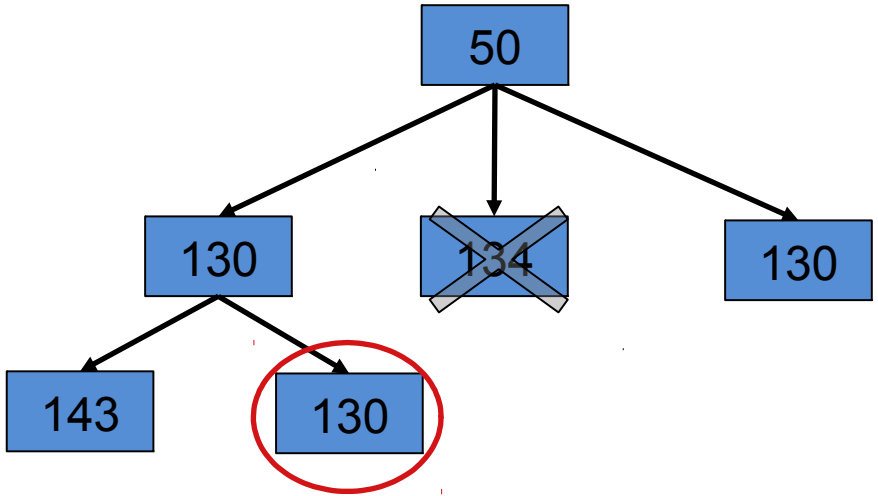


SWEDISH ICT SICS

Mathematical Programming

Enumerative algorithms

Branch and bound:

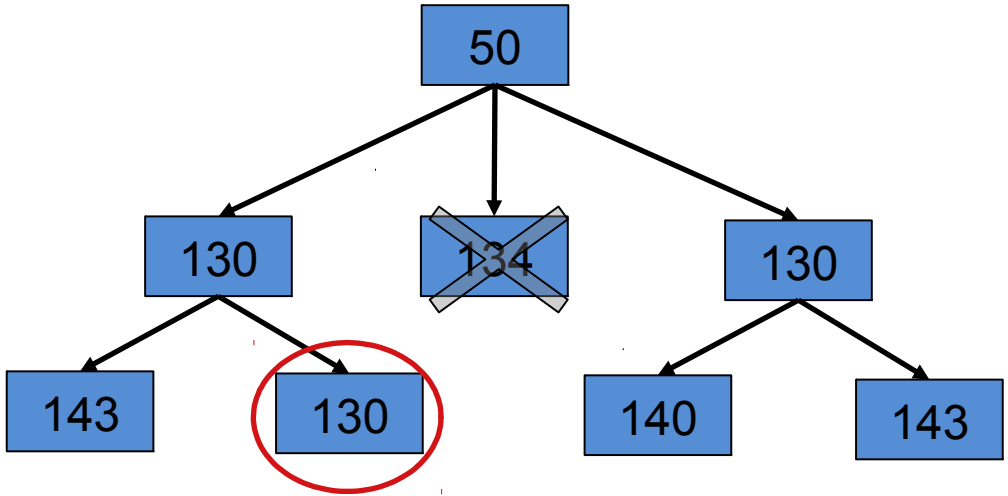


SWEDISH ICT SICS

Mathematical Programming

Enumerative algorithms

Branch and bound:



SWEDISH ICT SICS

Mathematical Programming

Enumerative algorithms

Dynamic Programming:

Bellman's Principle is a necessary condition:

"The sub-policy of an optimal policy is in itself optimal with regard to the start and end states"

Mathematical Programming

Enumerative algorithms

Dynamic Programming:

1. Divide the problem into stages *N*. For each stage there should be a number of states *x*.
2. Define the cost for the initial states in the initial stage.
3. Define a *recursive relation* for a state at stage *k* given states of previous stages.

$$x_{k+1} = f_k(x_k, u_k)$$

Mathematical Programming

Enumerative algorithms

Dynamic Programming:

Example: The knapsack problem.

1. Stages = items, states = how much weight capacity to use at this and all preceding stages, i.e. for this an all preceding items (0,1,2,3,4,5).
2. Cost of initial state, $f_3 = 30y_3$
3. Recursive relation:

$$f_i(y_i) = max_{k_i \leq \left\lfloor \frac{y_i}{w_i} \right\rfloor} \{u_i k_i + f_{i+1}(y_i - w_i k_i)\}$$

Item	Weight (w)	Utility (u)
1	2	65
2	3	80
3	1	30

SWEDISH ICT SICS

Relaxation and decomposition methods

Linear programming relaxation:

- Ignore the integrality constraint of an IP.
- Handy for finding lower bounds.

Mathematical Programming

Relaxation and decomposition methods

Lagrangian relaxation:

- Move a constraint into the objective function (and weigh it with λ)

$$\begin{aligned} min\ \ & c^T x \\ s.t.\ \ & A_1 x \leq b_1 \\ & A_2 x \leq b_2 \end{aligned}$$

$$\begin{aligned} min\ \ & c^T x + \lambda^T (b_2 - A_2 x) \\ s.t.\ \ & A_1 x \leq b_1 \end{aligned}$$

Relaxation and decomposition methods

Benders Decomposition:

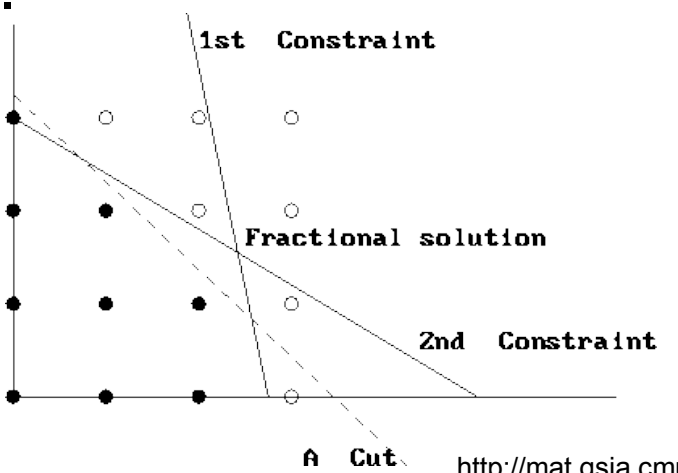
- A lot, but basically divide into master and sub-problem and use sub-problem to generate new constraints to master.

Mathematical Programming

Cutting plane and pricing algorithms

Cutting planes:

Add constraints to relaxed IP to generate tighter IP relaxations.



http://mat.gsia.cmu.edu/orclass/integer/node14.html

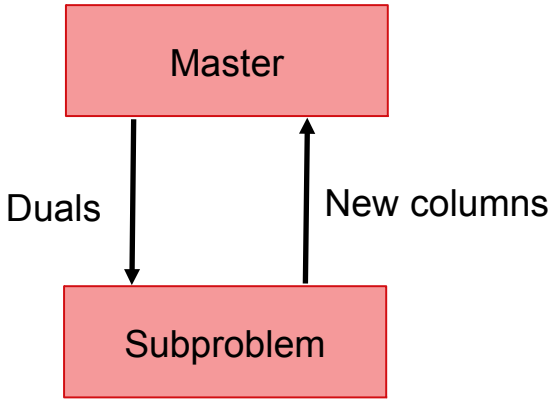
SWEDISH ICT SICS

Mathematical Programming

Cutting plane and pricing algorithms

Column generation (pricing):

Master problem (without all variables) and sub-problem (that generates variables that are likely to be useful to the Master).



SWEDISH ICT SICS

Low-level relay hybrid

S-metaheuristics in exact methods

Exact methods in S-metaheuristics

Low-level Relay Hybrid

S-metaheuristics in exact methods

Branch and Bound:

1. **Node selection strategy**: Which node should we branch on next? How? Use problem knowledge and S-metaheuristics.
2. **Upper bound generation**: Use S-metaheuristics to make complete solution from partial at nodes.

Cutting planes and Pricing algorithms:

2. Cut generation: Extract set of violated capacity constraints of the relaxed problem.
3. Generate new variables (columns): what it says.

SWEDISH ICT SICS

Low-level Relay Hybrid

Exact methods in S-metaheuristics

1. Very large neighbourhoods: use mathematical programming to search them and find the best or improving solution in the (whole or subset) neighbourhood (B&B, DP, Network Flow algorithms, matching algorithms).

2. Lower Bounds: use in S-metaheuristic. May also use other information e.g. Lagrangian multipliers.

Low-level Teamwork hybrid

P-metaheuristics in exact methods

Exact methods in P-metaheuristics

Low-level Teamwork Hybrid

P-metaheuristics in exact methods

Branching

- Which node to branch on? Which values to use in branching? Which node to solve next?
- Genetic algorithms have been used for solving the node selection problem.

Low-level Teamwork Hybrid

P-metaheuristics in exact methods

Local Branching (sort of like local search in MIP).

Local branching can be used for binary variables. Assumes *s* is a partial solution (of the binaries in the problem). Then let the optimization algorithm solve the problem for the k-opt neighbourhood of s, and if no better solution is found, solve it for the other branch (change k or more variables).

$$p_1 = \{x \epsilon \{0,1\}^n | \Delta(x,s) \leq k\} \qquad p_2 = \{x \epsilon \{0,1\}^n | \Delta(x,s) \geq k+1\}$$

SWEDISH ICT SICS

Low-level Teamwork Hybrid

Exact methods in p-metaheuristics

Recombination, mutation:

- Use an exact algorithm for combining parents into new children, mutation.

Exact decoding:

- Use exact algorithms for generating a solution from the incomplete encoding used by the heuristics.

Low-level Teamwork Hybrid

Exact methods in p-metaheuristics

Exact search ingredients:

- *Lower bounds:* Gilmore-Lawler lower bounds and dual variable values have been used in the construction phase of ant colony to solve the quadratic assignment problem. Also, lower bounds have been used in EA for mutation/crossover, where solutions that exceed a given bound are deleted.

- *Partial solutions:* The partial solutions of Branch and Bound may be interesting initial solutions.

High-level relay hybrid

Information provided by metaheuristics

Information provided by exact methods

Information provided by metaheuristics

1. **Good upper bound**: Solve the problem using a heuristic and start from that heuristic. Helps you prune more.

2. **Fixing variables**: Partition variables into set X and Y, let metaheuristic fix variables in set X and solve for variables in Y using exact methods.

3. **Domain reduction**: Heuristically perform a domain reduction for the decision variables and solve exactly within the reduced domain.

High-level Relay Hybrid

Information provided by exact algorithms

1. **Partial solutions**: Complete with metaheuristic.

2. **Problem reduction**: Used tree-search to reduce scheduling problem size and then used tabu search with simplified objective function to fins schedule.

3. **Relaxed optimal solution and their duals**: May be exploited by metaheuristics.

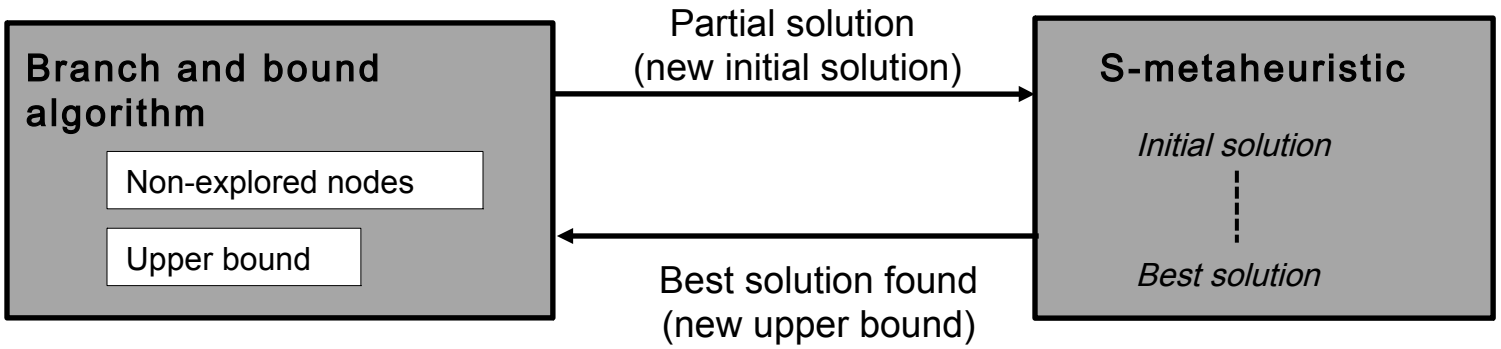
High-level teamwork hybrid

Parallel cooperation

Specialist cooperation

High-level Teamwork Hybrid

Parallel Cooperation

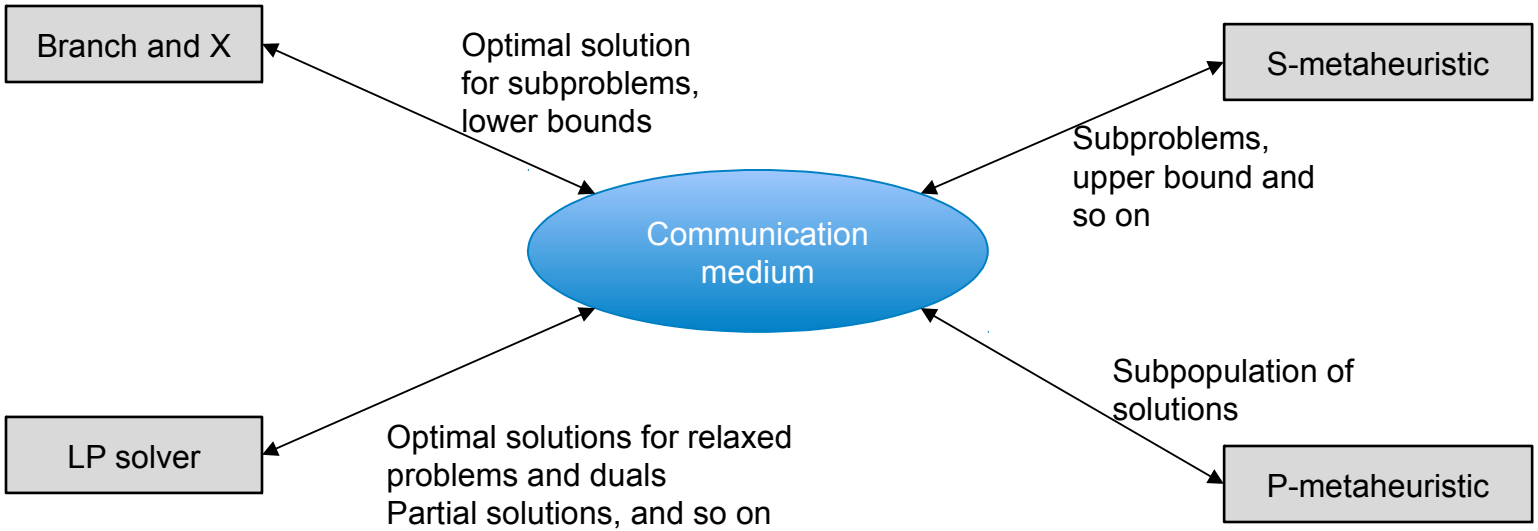


E.g. combining Branch and Bound with Simulated Annealing. The SA algorithm send improved upper bounds to B&B, and any integer solution found by B&B is used as an alternative reheated solution.

SWEDISH ICT SICS

High-level teamwork hybrid

Specialist cooperation



E.g. local search used to generate new columns.

SWEDISH ICT SICS

5.2 Combining metaheuristics with Constraint Programming

Constraint Programming

Hybrids

Constraint Programming

CP vs MH

Propagation and search

Constraint programming vs. Metaheuristics

Constraint Programming: good at solving combinatorial optimization problems that are tightly constrained.

Metaheuristics: good at solving problems that are underconstrained.

Constraint programming

Search and propagate

Constraints: Constraint programs consists of a set of variables linked by a set of constraints, e.g. the all_different constraint. The problem is solved by iterating between:

1. **Propagate:** Remove all values from variable domains that can't be in a feasible solution. There's lot's of code for this already.

2. **Search:** Tree search, partition problem into sub-problems by adding new constraints. You'll probably need to write your own code for this.
    1. **Branch ordering?**
    2. **Variable selection?**

SWEDISH ICT SICS

Hybrids

Low-level relay hybrid
Low-level teamwork hybrid
High-leve relay hybrid
High-level teamwork hybrid

Hybrids

Low-level relay hybrid

- **Neighbourhoods with expensive testing of feasibility:** CP is very good at feasibly!
- **Large neighbourhoods:** Once again, CP is good at feasibility…?

Low-level teamwork hybrids

Embedding metaheuristics into Constraint Programming:

- **Node improvement:** Use metaheuristics to improve or repair the nodes of the search tree.
- **Discrepancy-based search algorithms:** Generate near-greedy paths in a serch tree.
- **Branch-ordering:** Use metaheuristic to decide which child node to investigate first.
- **Variable selection:** Choose which branches to make.
- **Branching restriction:** Filter the branches of the search tree.

Low-level teamwork hybrids

Embedding constraint programming into metaheuristics:

- Recombination, large neighbourhoods, lower bounds, partial solutions, decoders…

High-level relay hybrids

Information provided by metaheuristic:

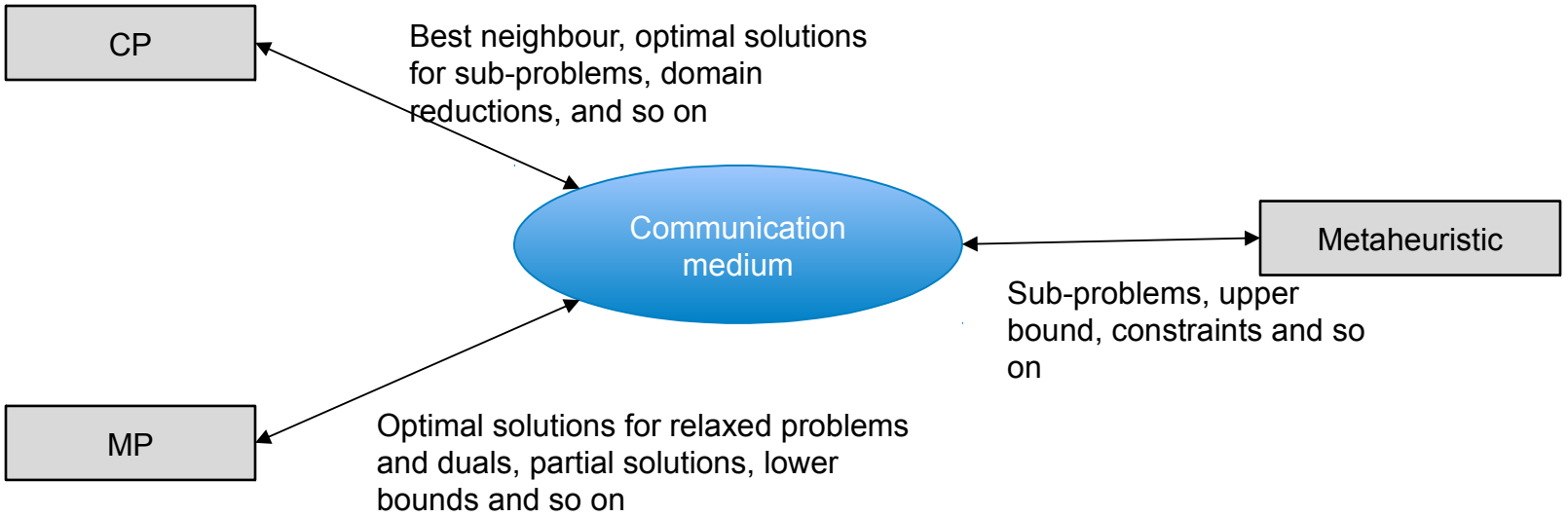
- Same as for when combining with MP.

Information provided by CP:

- Same as for when combining with MP.

Hybrids

High-level teamwork hybrids



Not that many.

SWEDISH ICT SICS

www.sics.se

PART OF RI.SE

SWEDISH ICT SICS