Adaptive propagation for efficient solving

Anastasia Paparrizou

Constraint Programming (CP)

CP solver =

backtrack search +

variable selection +

constraint propagation

Constraint propagation =

Enforce levels of consistency to reduce the search space

- remove infeasible values

- CP solvers usually apply one propagation level (the standard is Arc Consistency)

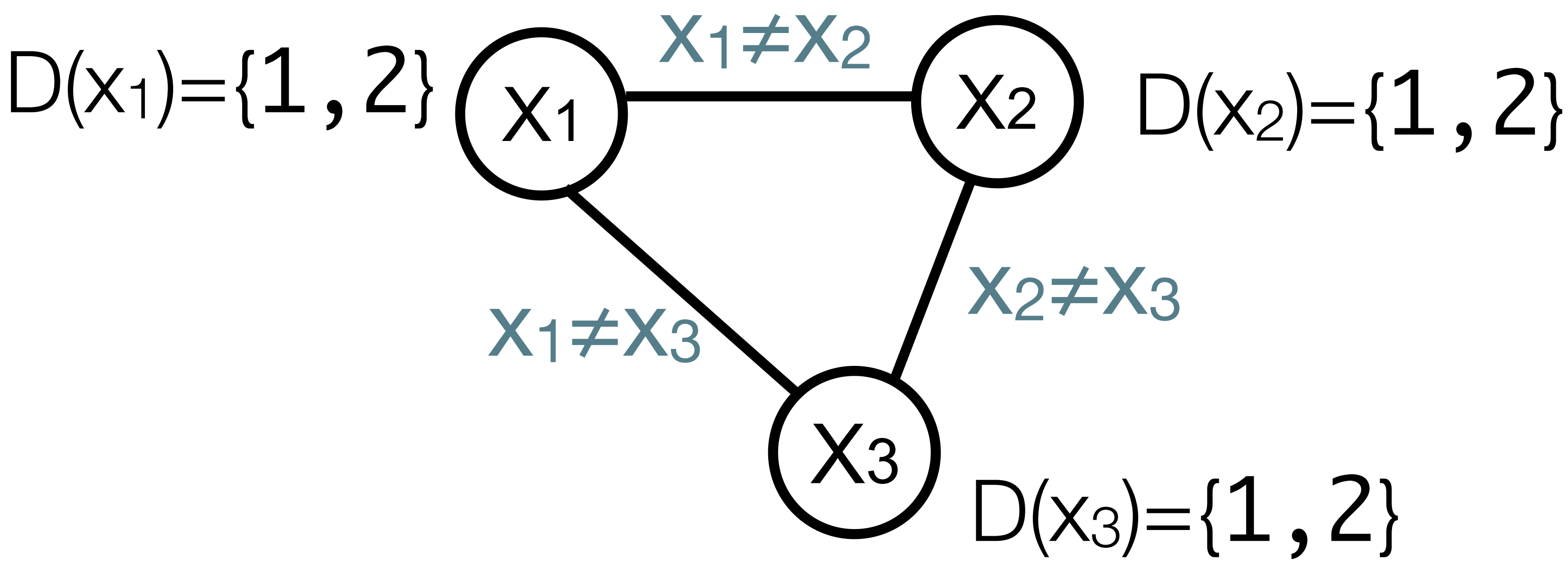
Why more than Arc Consistency?

Why not only one level of consistency?

When and how to change the consistency level?

Why more than Arc Consistency?

When AC is not enough to detect inconsistency



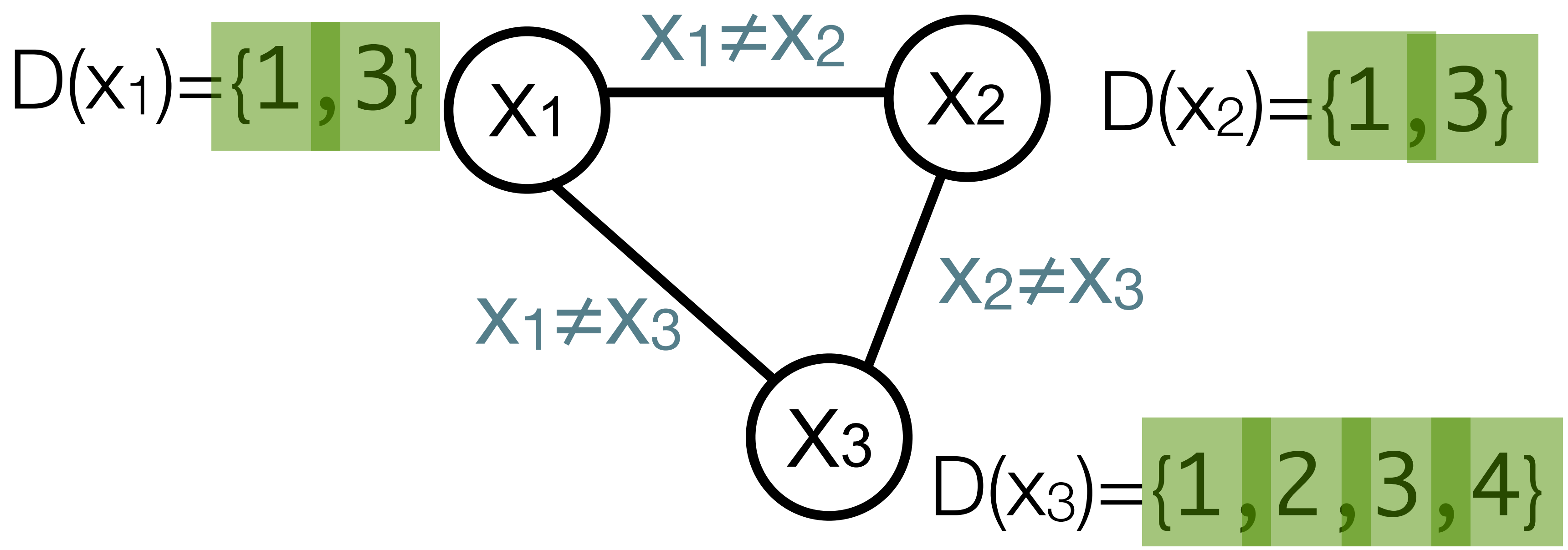
- It is AC…but UNSAT !
- Any consistency that considers triangles can detect this inconsistency (i.e., PIC, PC)
- If there exists a clique of disequalities (or anything else) on x_1 . . . x_k and we enforce (1,k-1)-consistency we will detect inconsistency.

AC is not enough when decomposing global cons

- Apply AC on a global constraint:

alldiff (x_1, x_2, x_3)

x_1	x_2	x_3
1	1	~~1~~
		2
3	3	~~3~~
		4



- Many global constraints are not AC decomposable.
- Solvers don't have an ad-hoc AC algorithm for each global constraint

Strong consistencies exist

- Path Consistency [Montanari ’74]
- k-Consistency [E. Freuder '78]
- Restricted Path Consistency [P. Berlandier ’95]
- Path Inverse Consistency [E. Freuder and C. Elfe ’96]
- Neighborhood Inverse Consistency [E. Freuder and C. Elfe ’96]
- max Restricted Path Consistency [R. Debruyne and C. Bessiere ’97]
- Singleton Arc Consistency [R. Debruyne and C. Bessiere ’97]

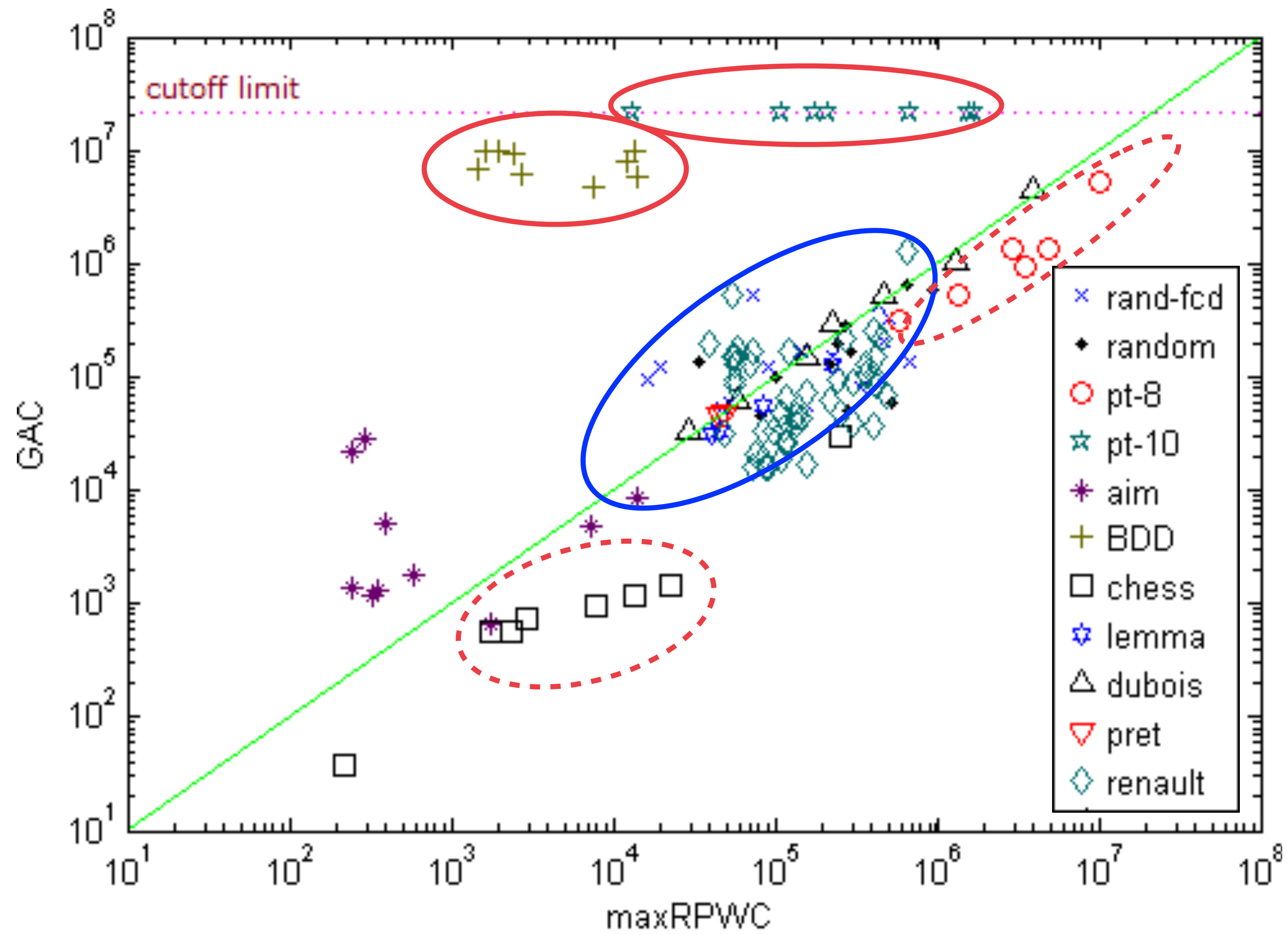
Why not only one level of consistency?

Theoretical viewpoint - tractability

- Some classes of problems are tractable under a specific level of consistency (not only AC).
  - max-closed constraints are AC decidable
  - row-convex constraints are PC decidable
  - majority polymorphisms are SAC decidable
- BUT:
  - Usually they don’t appear purely in practice

Practical viewpoint - Illustration

- Different classes of CP problems respond best to different consistency algorithms.



- Some problems would benefit from several propagation levels.
- The user cannot decide the right level.

- The solver need to revise its behavior at **runtime** depending on its performance **during search**.

When and how to change the consistency level?

Adapting dynamically the level of consistency

IJCAI'15

Related works

- Machine learning was used off-line
  - especially for variable selection.
- Heuristic methods require tuning of parameters
  - defined only for 2 levels of consistency.
- Portfolio approaches need training to select the solver/ parameters for classes of problems.
  - no feedback during search.

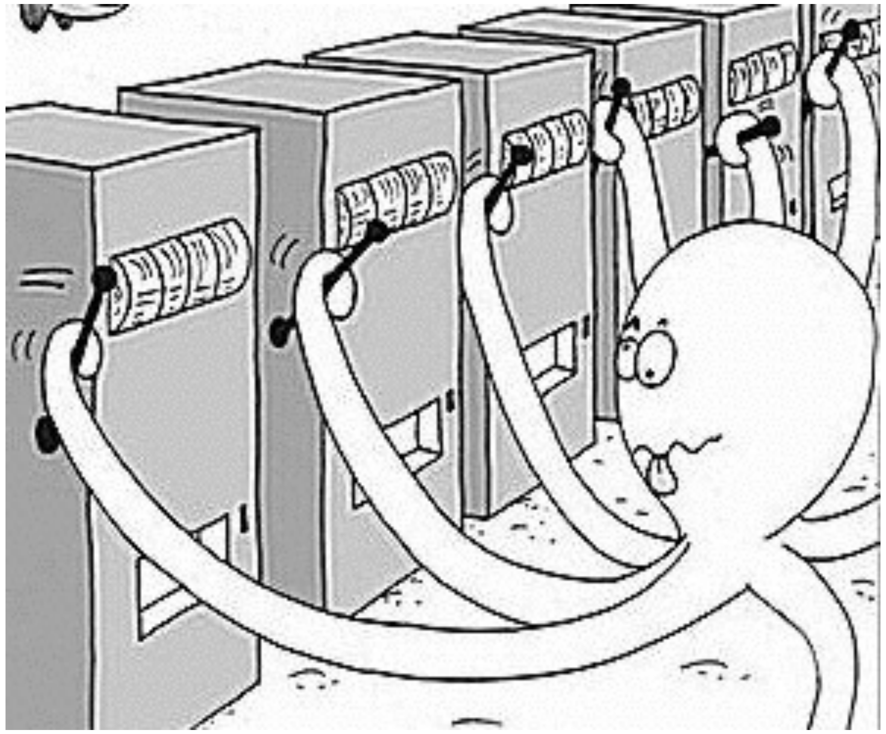
A new machine learning approach

- We use multi-armed bandits to decide the appropriate level of consistency.
  - Without parameters that require tuning
  - Without offline/static learning
  - With a general criterion as a reward function
- General framework for any number of consistencies.

The Multi-armed bandit (MAB) model

- A set of ***k* arms** $\{LC_1, \ldots, LC_k\}$. Each arm corresponds to an algorithm that enforces a specific level of local consistency.

- A **reward** delivered when an arm LC_i has been selected.



➡ GOAL: select from a set of LCs in a sequence of trials so as to maximize the total payoff.

Exploitation vs Exploration

- Trade-off between acquiring new information (exploration) and capitalizing on the information available (exploitation).
- *Upper confidence bound* (UCB) ensures that no arm will be forgotten.

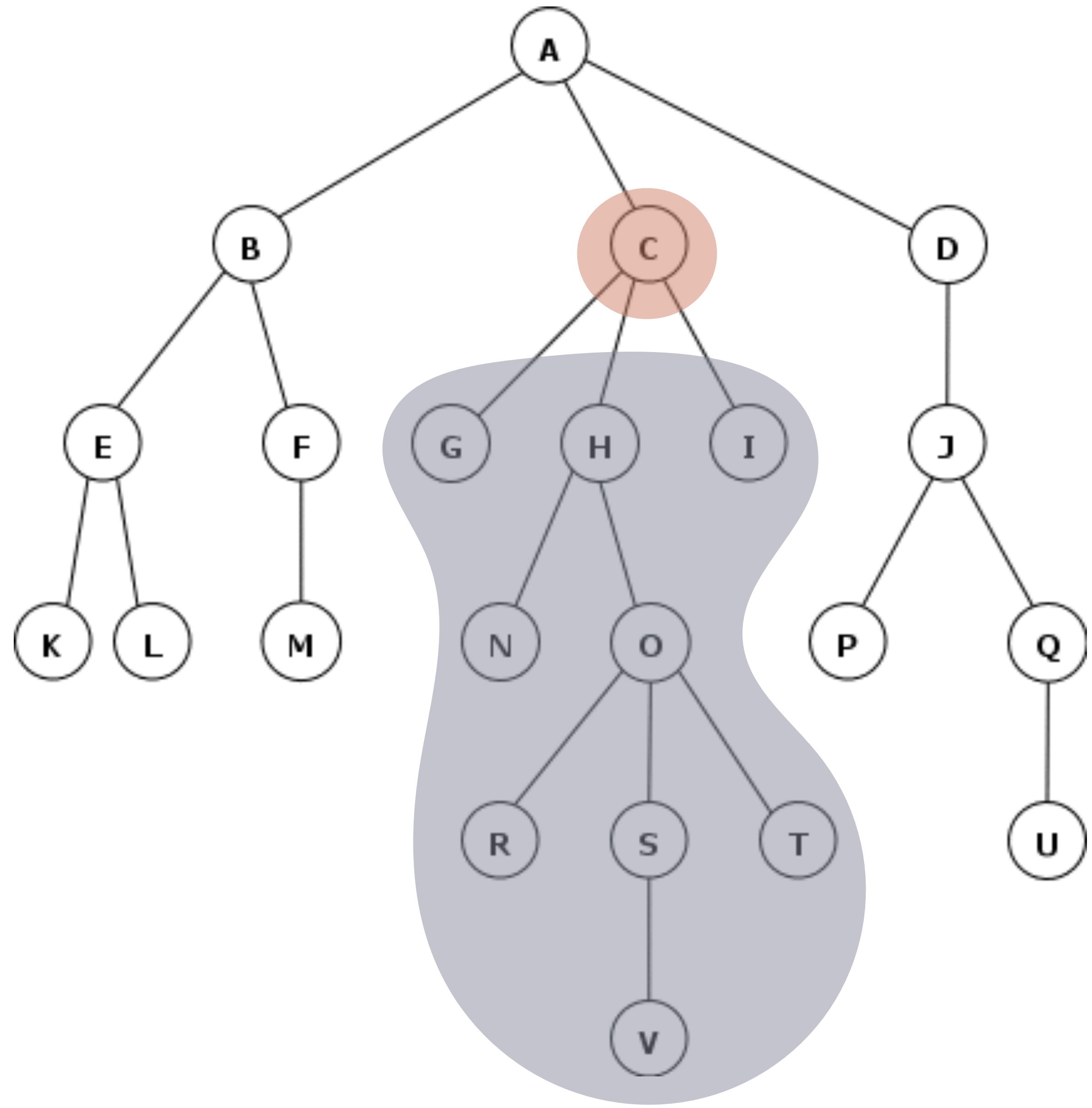
The MAB model (cont'd)

- One MAB per depth
- Reward based on:

  the CPU time needed to enforce LC_i on a node

  +

  the time to explore the subtree rooted at this node.



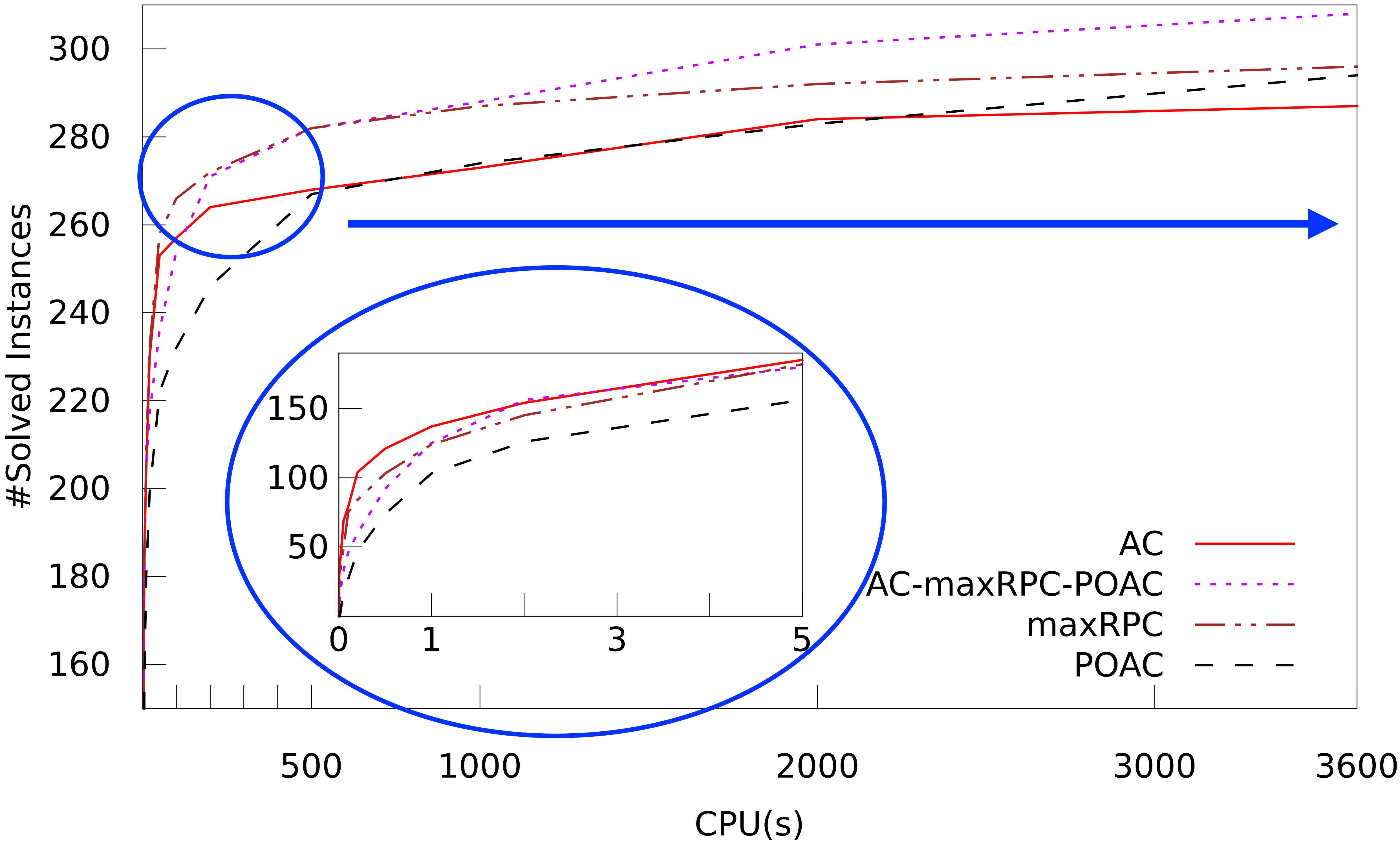
Algorithm

When MAB is applied per depth:

1. We call the MAB selector of the depth at which **x←a** occurs.
2. We select the LC_i that maximizes **ρ(i)**.
3. We store the current time **startTime[depth]** of the machine.
4. LC_i is executed on that node.
5. When backtracking to that node, we update the reward: **$T_i(j)$** = CPU time to enforce LC_i + CPU time to explore the resulting subtree = **endTime – startTime[depth]**.

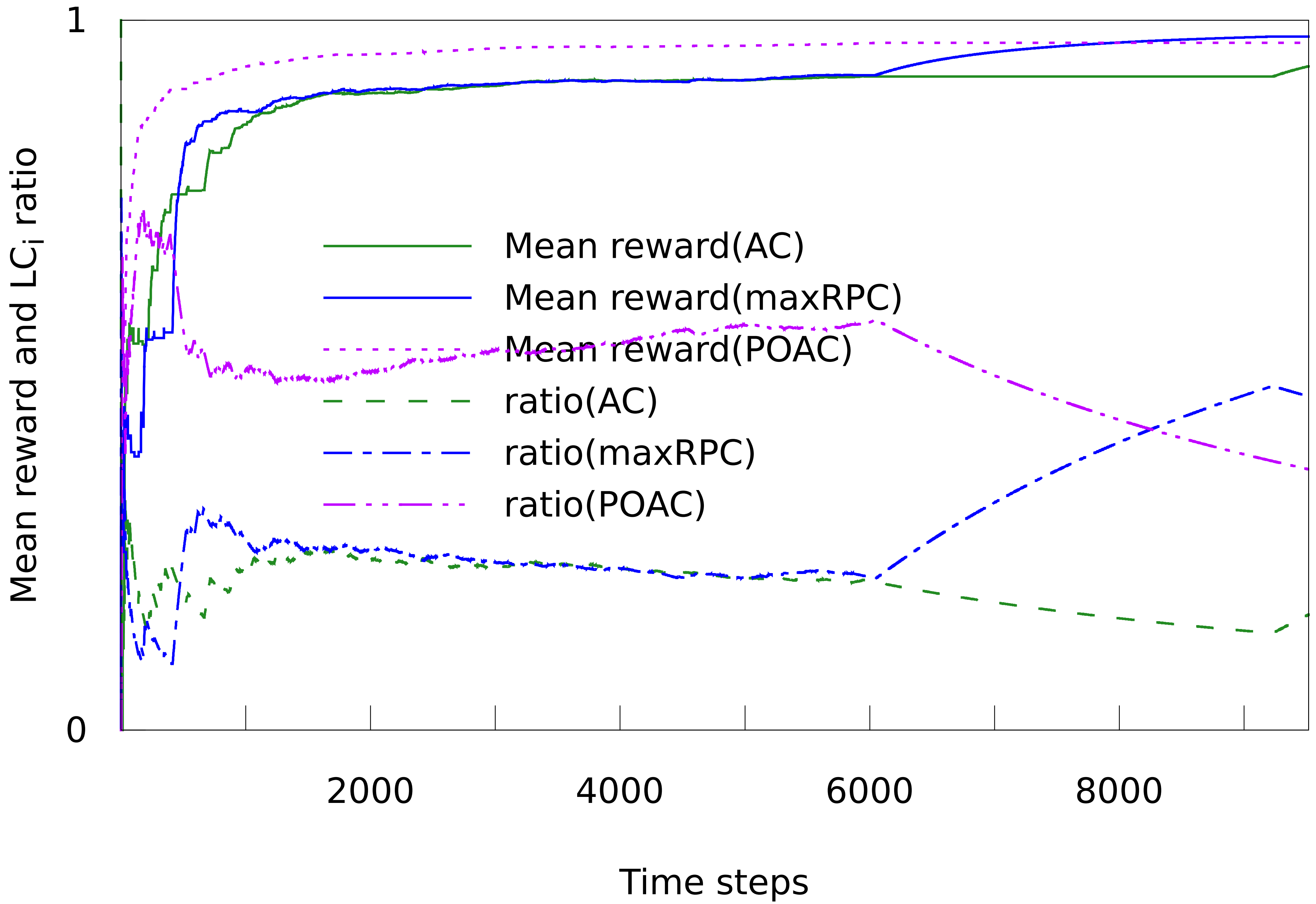
Evaluation:

Number of solved instances



Evaluation:

3-insertion-4-3 (depth 104)



Sum up

- We introduced a general framework for adaptive constraint propagation based on MAB learning.
  - Automatic selection of the right level of propagation among several levels.
  - Light learning mechanism, it can be applied dynamically, considering the effects of propagation during search.
- Increases the efficiency and robustness of a CP solver.