Best Heuristic Identification for Constraint Satisfaction

Frederic Koriche, Christophe Lecoutre, Anastasia Paparrizou, **Hugues Wattez**

11 octobre 2022

Présentation à la 6ème journée CAVIAR

Plan

Introduction

Constraint Satisfaction Problem

Definition (Variable)

A *variable* x is an entity associated to a value. This value belongs to its *domain*, denoted $\texttt{dom}(x)$.

Definition (Constraint)

A *constraint* c is defined by a set of variables, called *scope* of c and denoted $\texttt{scp}(c)$, and by a mathematical relation which describes the set of tuples allowed by c for the variables of its scope.

Constraint Satisfaction Problem

Definition (Variable)

A *variable* x is an entity associated to a value. This value belongs to its *domain*, denoted $\mathtt{dom}(x)$.

Definition (Constraint)

A *constraint* c is defined by a set of variables, called *scope* of c and denoted $\mathtt{scp}(c)$, and by a mathematical relation which describes the set of tuples allowed by c for the variables of its scope.

Constraint Satisfaction Problem

Definition (CSP)

A *Constraint Satisfaction Problem* (or *Constraint Network*) $\mathcal{P}$ is defined by:

- a finite set of **variables**, denoted $\mathcal{X}$
- a finite set of **constraints**, denoted $\mathcal{C}$, such that $\forall c \in \mathcal{C}, \texttt{scp}(c) \subseteq \mathcal{X}$

Definition (Solution)

A *solution* of a CSP instance $\mathcal{P}$ corresponds to the assignment of a value to each variable of $\mathcal{X}$ such that all the constraints of $\mathcal{C}$ are satisfied.

Definition (CSP)

A *Constraint Satisfaction Problem* (or *Constraint Network*) $\mathcal{P}$ is defined by:

- a finite set of **variables**, denoted $\mathcal{X}$
- a finite set of **constraints**, denoted $\mathcal{C}$, such that $\forall c \in \mathcal{C}, \mathtt{scp}(c) \subseteq \mathcal{X}$

Definition (Solution)

A *solution* of a CSP instance $\mathcal{P}$ corresponds to the assignment of a value to each variable of $\mathcal{X}$ such that all the constraints of $\mathcal{C}$ are satisfied.

Solving Principle of a Constraint Solver

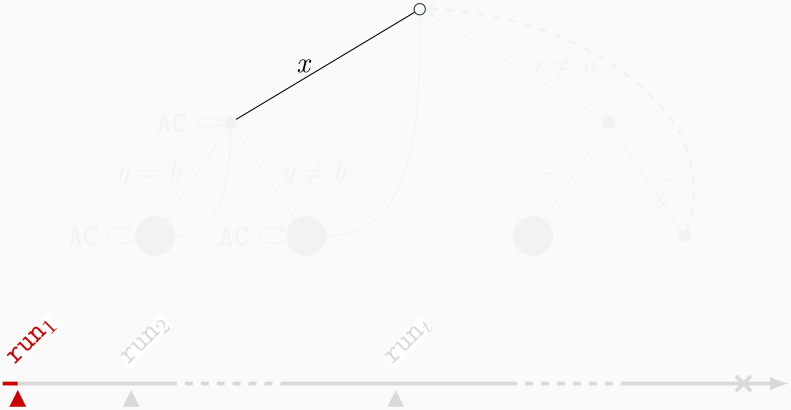
Global scheme : depth first binary tree search with backtracking

Solving Principle of a Constraint Solver

1^{st} **run** : root of the tree

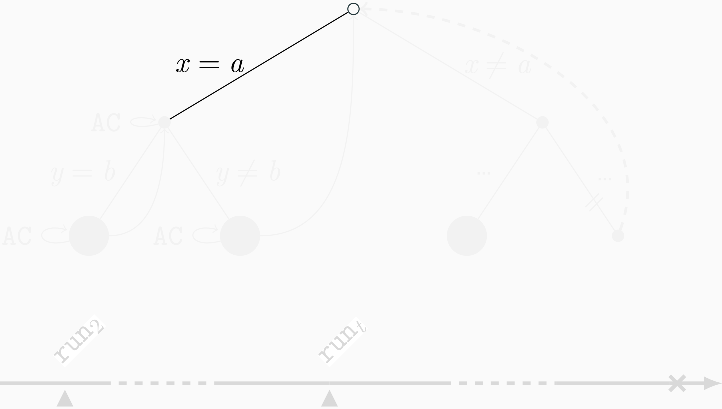
Solving Principle of a Constraint Solver

Decision : the variable ordering heuristic selects x



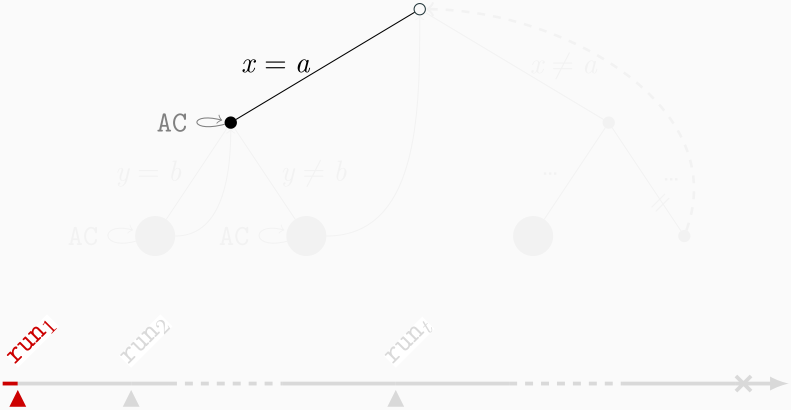
Solving Principle of a Constraint Solver

Decision : the value ordering heuristic selects a



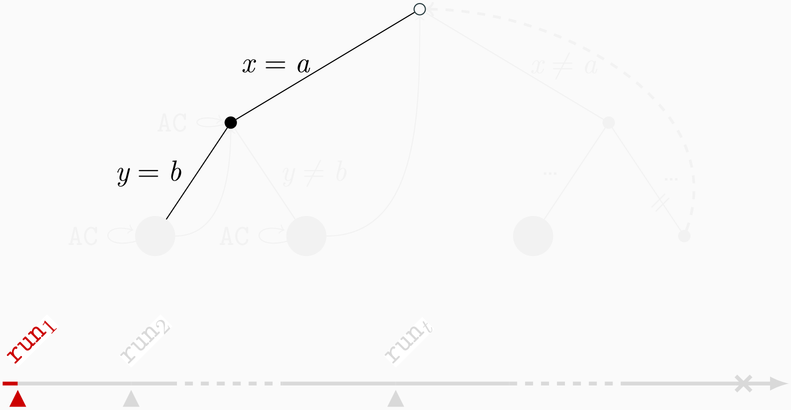
Solving Principle of a Constraint Solver

Propagation : enforcing of the arc-consistency (AC) property



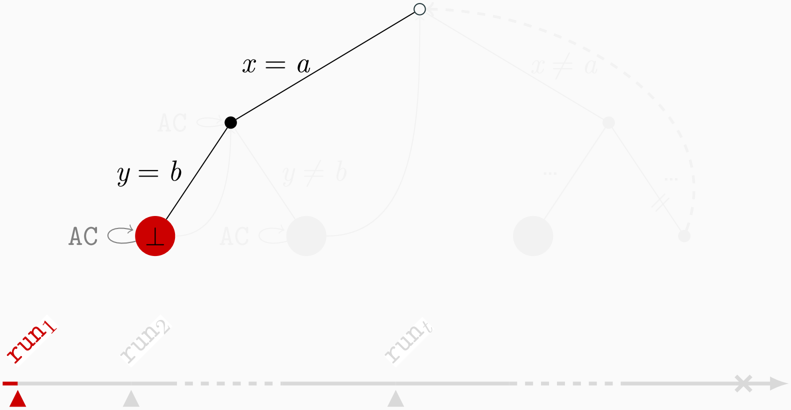
Solving Principle of a Constraint Solver

Decision : next selection (variable, valeur)



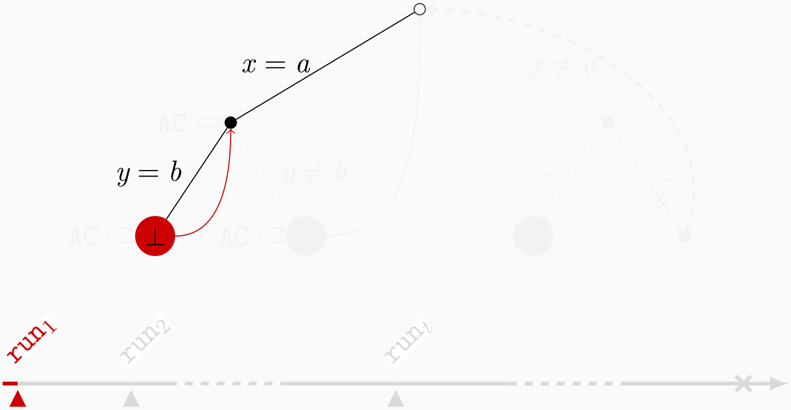
Solving Principle of a Constraint Solver

Propagation : enforcing of the AC property and conflict



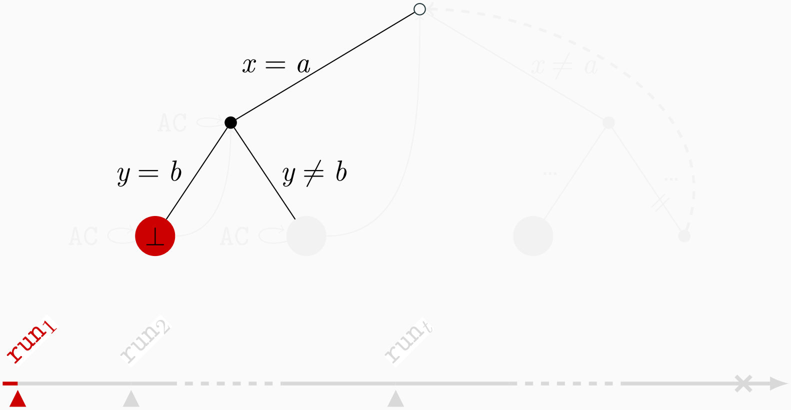
Solving Principle of a Constraint Solver

Backtracking : parent node



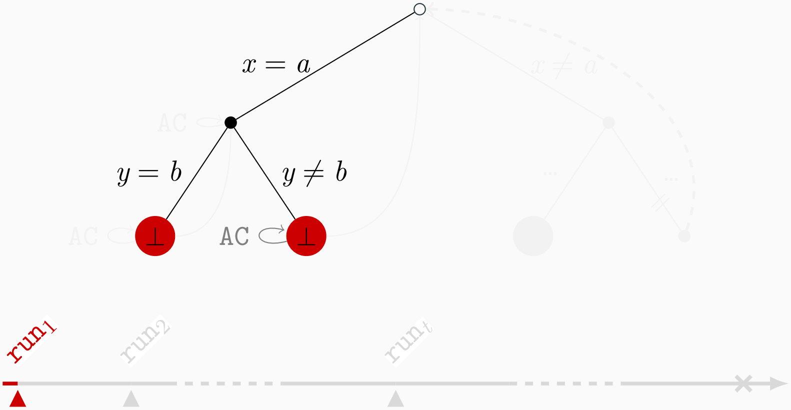
Solving Principle of a Constraint Solver

Refutation : we consider $y \neq b$



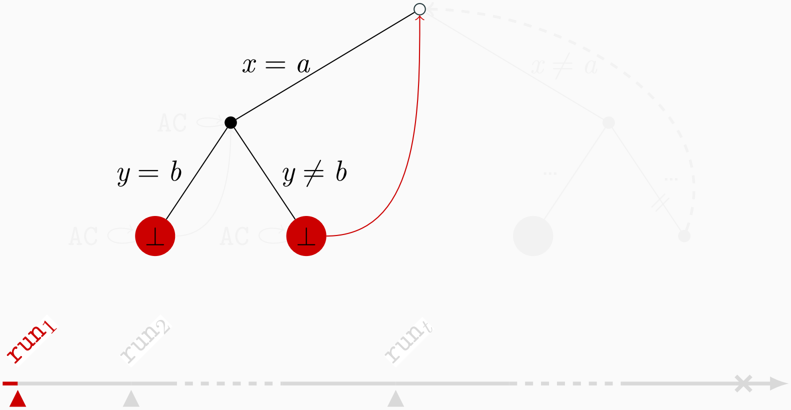
Solving Principle of a Constraint Solver

Propagation : enforcing of the AC property and conflict



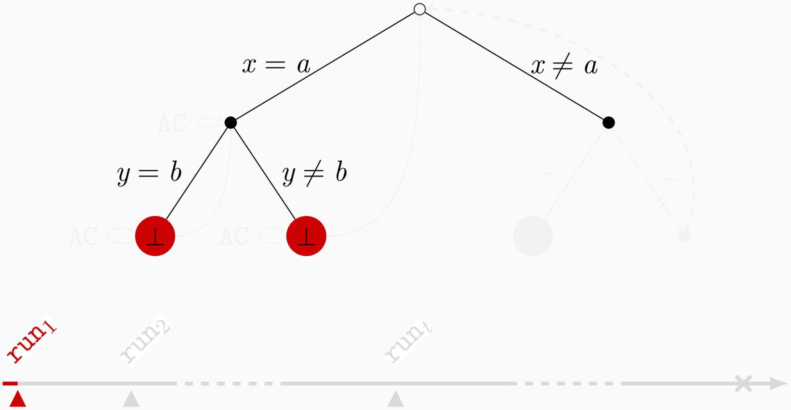
Solving Principle of a Constraint Solver

Backtracking : parent node (root node)



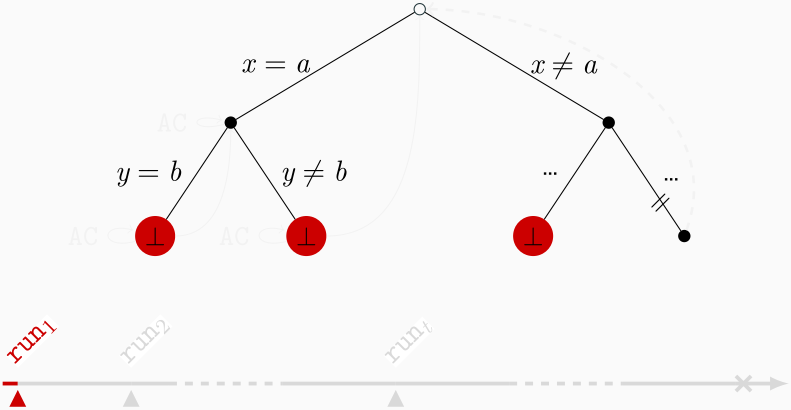
Solving Principle of a Constraint Solver

Refutation : $x \neq a$



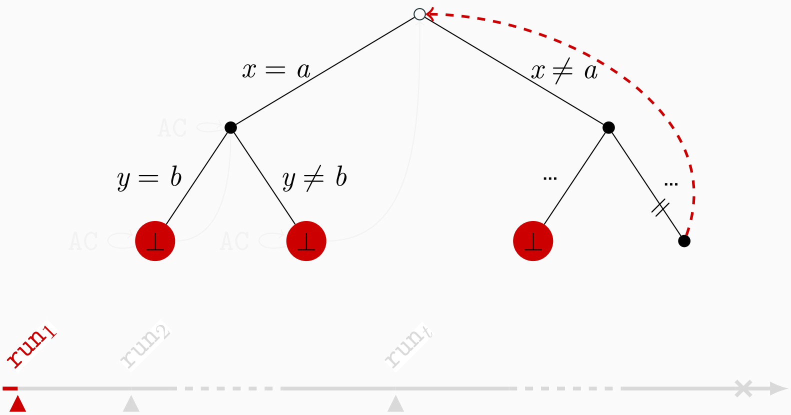
Solving Principle of a Constraint Solver

Restarting : cutoff reached and nogood extraction



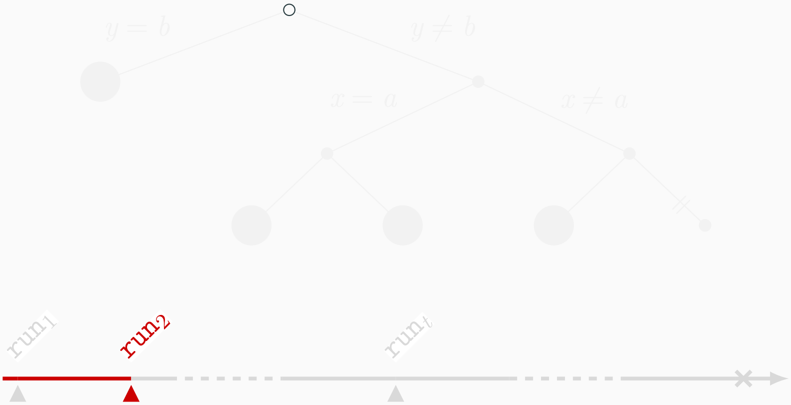
Solving Principle of a Constraint Solver

Restarting : backtrack to the root node



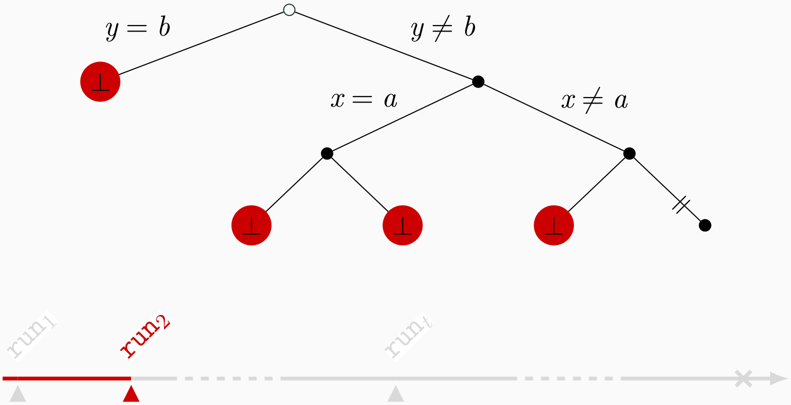
Solving Principle of a Constraint Solver

2^{nd} **run** : root node



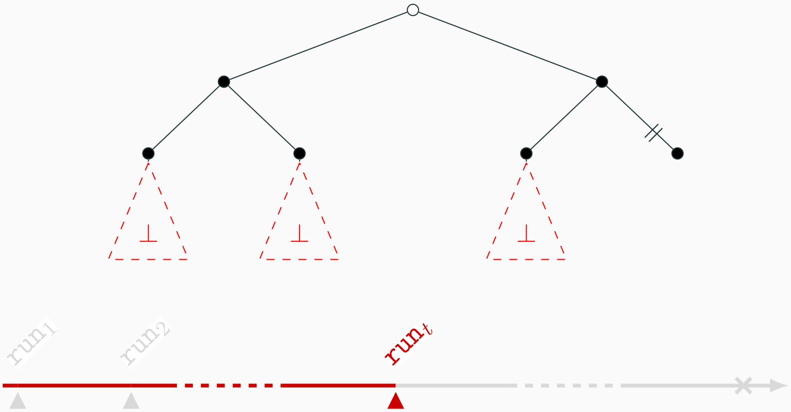
Solving Principle of a Constraint Solver

2^{nd} **run** : cutoff reached and restarting



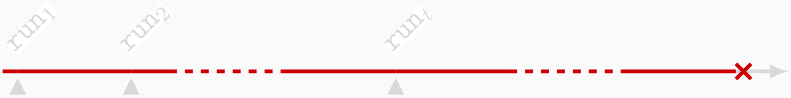
Solving Principle of a Constraint Solver

$\mathbf{t^{th}}$ **run** : cutoff reached and restarting



Solving Principle of a Constraint Solver

End of solving : satisfiability | unsatisfiability | timeout



Variable ordering heuristics

Let the set $\mathcal{H} = \{\texttt{lex}, \texttt{dom}, \texttt{dom/ddeg}, \texttt{abs}, \texttt{ibs}, \texttt{dom/wdeg}, \texttt{chs}, \texttt{cacd}\}$:

`lex`: lexicographic order

Variable ordering heuristics

Let the set $\mathcal{H} = \{\texttt{lex}, \texttt{dom}, \texttt{dom/ddeg}, \texttt{abs}, \texttt{ibs}, \texttt{dom/wdeg}, \texttt{chs}, \texttt{cacd}\}$:

`lex`: lexicographic order

`dom`: size of domain

Variable ordering heuristics

Let the set $\mathcal{H} = \{\texttt{lex}, \texttt{dom}, \texttt{dom/ddeg}, \texttt{abs}, \texttt{ibs}, \texttt{dom/wdeg}, \texttt{chs}, \texttt{cacd}\}$:

`lex`: lexicographic order

`dom`: size of domain

`dom/ddeg`: size of domain and variable degree

Variable ordering heuristics

Let the set $\mathcal{H} = \{\texttt{lex}, \texttt{dom}, \texttt{dom/ddeg}, \texttt{abs}, \texttt{ibs}, \texttt{dom/wdeg}, \texttt{chs}, \texttt{cacd}\}$:

`lex`: lexicographic order

`dom`: size of domain

`dom/ddeg`: size of domain and variable degree

`abs`: activity of variables

Variable ordering heuristics

Let the set $\mathcal{H} = \{\texttt{lex}, \texttt{dom}, \texttt{dom/ddeg}, \texttt{abs}, \texttt{ibs}, \texttt{dom/wdeg}, \texttt{chs}, \texttt{cacd}\}$:

- `lex`: lexicographic order
- `dom`: size of domain
- `dom/ddeg`: size of domain and variable degree
- `abs`: activity of variables
- `ibs`: impact of variables

Variable ordering heuristics

Let the set $\mathcal{H} = \{\texttt{lex}, \texttt{dom}, \texttt{dom/ddeg}, \texttt{abs}, \texttt{ibs}, \texttt{dom/wdeg}, \texttt{chs}, \texttt{cacd}\}$:

$\texttt{lex}$: lexicographic order

$\texttt{dom}$: size of domain

$\texttt{dom/ddeg}$: size of domain and variable degree

$\texttt{abs}$: activity of variables

$\texttt{ibs}$: impact of variables

$\texttt{dom/wdeg}$: size of domain and constraint weighting

Variable ordering heuristics

Let the set $\mathcal{H} = \{\texttt{lex}, \texttt{dom}, \texttt{dom/ddeg}, \texttt{abs}, \texttt{ibs}, \texttt{dom/wdeg}, \texttt{chs}, \texttt{cacd}\}$:

- `lex`: lexicographic order
- `dom`: size of domain
- `dom/ddeg`: size of domain and variable degree
- `abs`: activity of variables
- `ibs`: impact of variables
- `dom/wdeg`: size of domain and constraint weighting
- `chs`: history of constraint conflicts

Variable ordering heuristics

Let the set $\mathcal{H} = \{\texttt{lex}, \texttt{dom}, \texttt{dom}/\texttt{ddeg}, \texttt{abs}, \texttt{ibs}, \texttt{dom}/\texttt{wdeg}, \texttt{chs}, \texttt{cacd}\}$:

- $\texttt{lex}$: lexicographic order
- $\texttt{dom}$: size of domain
- $\texttt{dom}/\texttt{ddeg}$: size of domain and variable degree
- $\texttt{abs}$: activity of variables
- $\texttt{ibs}$: impact of variables
- $\texttt{dom}/\texttt{wdeg}$: size of domain and constraint weighting
- $\texttt{chs}$: history of constraint conflicts
- $\texttt{cacd}$: arity and domain of the conflict variables

Problem

Heuristic determines search efficiency...

#instances	dom/wdeg	activity	impact
KnightTour	4	3	**5**
MultiKnapsack	24	**27**	25
Subisomorphism	**7**	2	5

Table 1: Solved instances by heuristic

... but heuristic selection needs expert qualities.

Given a CSP instance and a set of heuristics available in the solver, which heuristic is the best for solving the instance?

Problem

Heuristic determines search efficiency...

#instances	dom/wdeg	activity	impact
KnightTour	4	3	**5**
MultiKnapsack	24	**27**	25
Subisomorphism	**7**	2	5

Table 1: Solved instances by heuristic

... but heuristic selection needs expert qualities.

Given a CSP instance and a set of heuristics available in the solver, which heuristic is the best for solving the instance?

Problem

Heuristic determines search efficiency...

#instances	dom/wdeg	activity	impact
KnightTour	4	3	**5**
MultiKnapsack	24	**27**	25
Subisomorphism	**7**	2	5

Table 1: Solved instances by heuristic

... but heuristic selection needs expert qualities.

Given a CSP instance and a set of heuristics available in the solver, which heuristic is the best for solving the instance?

Multi-Armed Bandit Framework

Why this name?

One-armed bandits with different jackpot probabilities:

2% 1.5% 1% 1.5% 10%

The multi-armed bandit problem is characterized by:

- the search for a balance between exploration and exploitation

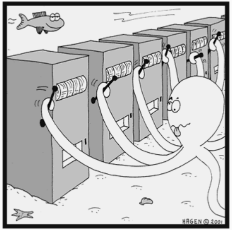
Why this name?

One-armed bandits with different jackpot probabilities:

2% 1.5% 1% 1.5% 10%

The multi-armed bandit problem is characterized by:

- the search for a balance between exploration and exploitation



Why this name?

One-armed bandits with different jackpot probabilities:

$$\begin{array}{ccccc} \mathcal{H}_1 & \mathcal{H}_2 & \mathcal{H}_3 & \mathcal{H}_4 & \mathcal{H}_5 \\ 2\% & 1.5\% & 1\% & 1.5\% & 10\% \end{array}$$

The multi-armed bandit problem is characterized by:

- the search for a balance between exploration and exploitation

Global description of a bandit

The bandit problem is described as a game where a player faces the environment. At each trial t:

- the player chooses an action i_t among a set of actions A (heuristic selection in our case)
- the environment gives a reward $r_t(i_t)$ to the player for the selected action

The player's goal is to minimize his regret after T trials:

$$Regret_T = \max_{i \in A} \sum_{t=1}^{T} r_t(i) - \sum_{t=1}^{T} r_t(i_t)$$

or, depending on the bandit paradigm, minimizing the number of trials required to explore before committing to an optimal arm.

Global description of a bandit

The bandit problem is described as a game where a player faces the environment. At each trial t:

- the player chooses an action i_t among a set of actions A (heuristic selection in our case)
- the environment gives a reward $r_t(i_t)$ to the player for the selected action

The player's goal is to minimize his regret after T trials:

$$Regret_T = \max_{i \in A} \sum_{t=1}^{T} r_t(i) - \sum_{t=1}^{T} r_t(i_t)$$

or, depending on the bandit paradigm, minimizing the number of trials required to explore before committing to an optimal arm.

Global description of a bandit

The bandit problem is described as a game where a player faces the environment. At each trial t:

- the player chooses an action i_t among a set of actions A (heuristic selection in our case)
- the environment gives a reward $r_t(i_t)$ to the player for the selected action

The player's goal is to minimize his regret after T trials:

$$Regret_T = \max_{i \in A} \sum_{t=1}^{T} r_t(i) - \sum_{t=1}^{T} r_t(i_t)$$

or, depending on the bandit paradigm, minimizing the number of trials required to explore before committing to an optimal arm.

Global description of a bandit

The bandit problem is described as a game where a player faces the environment. At each trial t:

- the player chooses an action i_t among a set of actions A (heuristic selection in our case)
- the environment gives a reward $r_t(i_t)$ to the player for the selected action

The player's goal is to minimize his regret after T trials:

$$Regret_T = \max_{i \in A} \sum_{t=1}^{T} r_t(i) - \sum_{t=1}^{T} r_t(i_t)$$

or, depending on the bandit paradigm, minimizing the number of trials required to explore before committing to an optimal arm.

Global description of a bandit

The bandit problem is described as a game where a player faces the environment. At each trial t:

- the player chooses an action i_t among a set of actions A (heuristic selection in our case)
- the environment gives a reward $r_t(i_t)$ to the player for the selected action

The player's goal is to minimize his regret after T trials:

$$Regret_T = \max_{i \in A} \sum_{t=1}^{T} r_t(i) - \sum_{t=1}^{T} r_t(i_t)$$

or, depending on the bandit paradigm, minimizing the number of trials required to explore before committing to an optimal arm.

Global description of a bandit

The bandit problem is described as a game where a player faces the environment. At each trial t:

- the player chooses an action i_t among a set of actions A (heuristic selection in our case)
- the environment gives a reward $r_t(i_t)$ to the player for the selected action

The player's goal is to minimize his regret after T trials:

$$Regret_T = \max_{i \in A} \sum_{t=1}^{T} r_t(i) - \sum_{t=1}^{T} r_t(i_t)$$

or, depending on the bandit paradigm, minimizing the number of trials required to explore before committing to an optimal arm.

Global description of a bandit

The bandit problem is described as a game where a player faces the environment. At each trial t:

- the player chooses an action i_t among a set of actions A (heuristic selection in our case)
- the environment gives a reward $r_t(i_t)$ to the player for the selected action

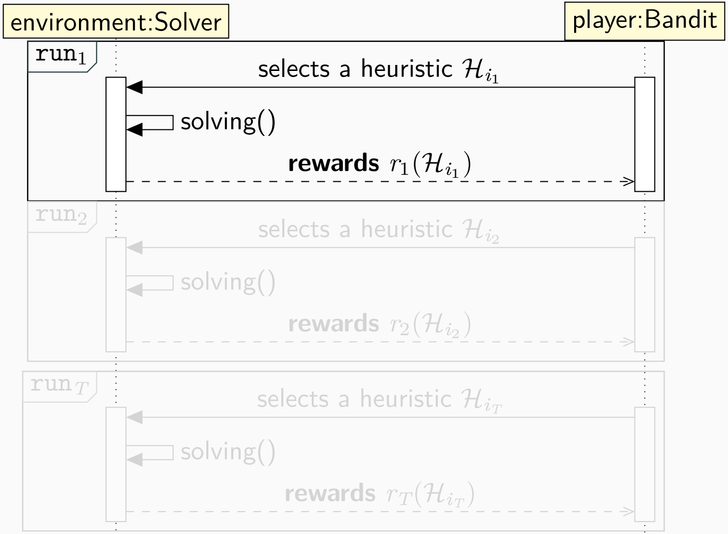
The player's goal is to minimize his regret after T trials:

$$Regret_T = \max_{i \in A} \sum_{t=1}^{T} r_t(i) - \sum_{t=1}^{T} r_t(i_t)$$

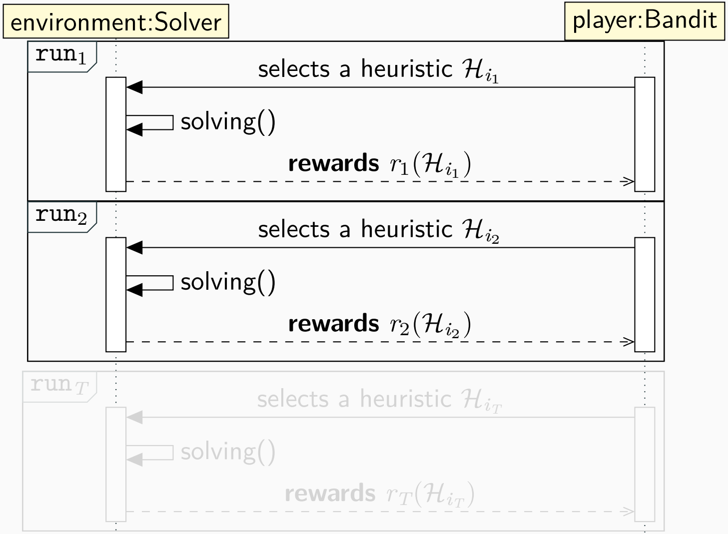
or, depending on the bandit paradigm, minimizing the number of trials required to explore before committing to an optimal arm.

Link between bandit and heuristics in a CSP solver

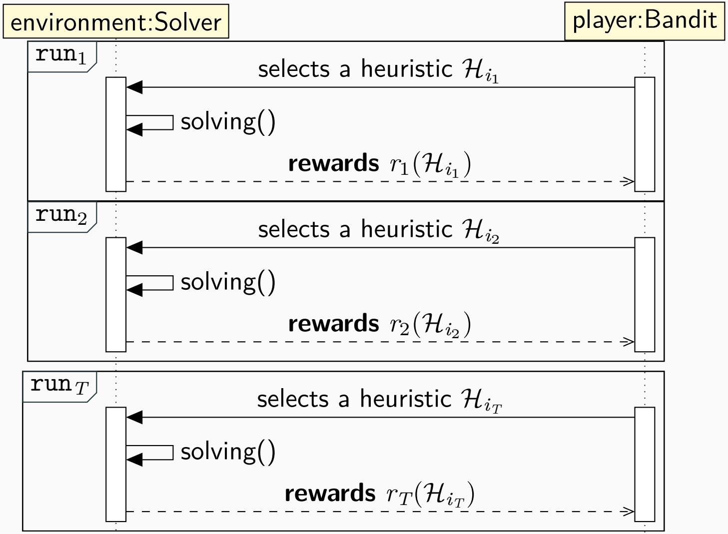
Link between bandit and heuristics in a CSP solver



Link between bandit and heuristics in a CSP solver



Link between bandit and heuristics in a CSP solver



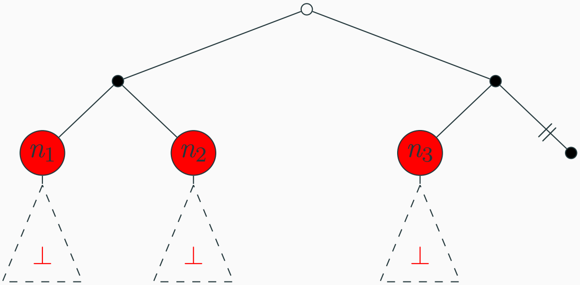
Reward function

The reward function is based on the size of the trees pruned during a `run` t:

$$r_t(i) = \frac{\log_2\left(\sum_{n \in \mathtt{cft}(\mathcal{T}_t)} \prod_{x \in \mathtt{fut}(n)} |\mathtt{dom}(x)|\right)}{\log_2\left(\prod_{x \in \mathtt{vars}(P)} |\mathtt{dom}(x)|\right)}$$

where:

- $\mathtt{cft}(\mathcal{T})$: the set of conflictual nodes
- $\mathtt{fut}(n)$: the set of unfixed variables
- $\mathtt{vars}(P)$: the variables of P
- $\mathtt{dom}(x)$: the domain of x



Selection policies

$\texttt{UCB}$: simple upper confidence bound (stochastic bandit)

$\texttt{EXP3}$: exponential weighting for exploration and exploitation (adversarial bandit)

$\texttt{UNI}$: random and uniform choice (naive policy)

$\texttt{VBS}$: virtual best solver (best policy)

Let the set of choice policies $\mathcal{B} = \{\texttt{UCB}, \texttt{EXP3}, \texttt{UNI}, \texttt{VBS}, \}$:

Selection policies

$\texttt{UCB}$: simple upper confidence bound (stochastic bandit)

$\texttt{EXP3}$: exponential weighting for exploration and exploitation (adversarial bandit)

$\texttt{UNI}$: random and uniform choice (naive policy)

$\texttt{VBS}$: virtual best solver (best policy)

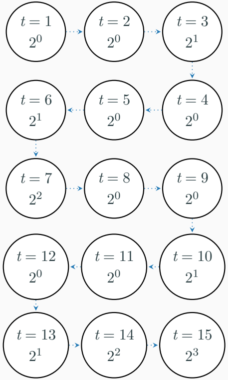
Let the set of choice policies $\mathcal{B} = \{\texttt{UCB}, \texttt{EXP3}, \texttt{UNI}, \texttt{VBS}, \texttt{AST}\}$:

Adaptive Single Tournament

From sequential Luby to arborescent Luby

(a) Sequential view (b) Tree-structured view

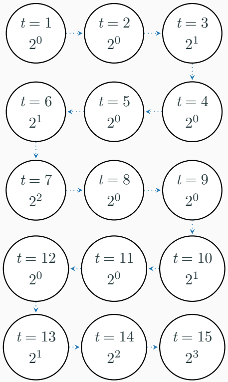
From sequential Luby to arborescent Luby



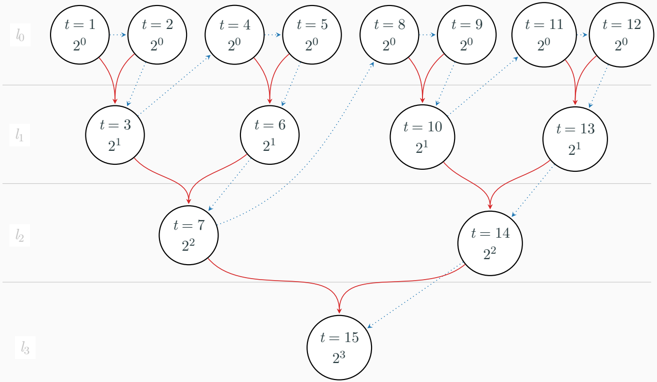
(a) Sequential view

(b) Tree-structured view

From sequential Luby to arborescent Luby

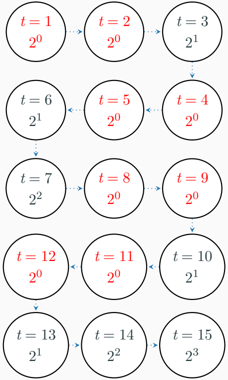


(a) Sequential view

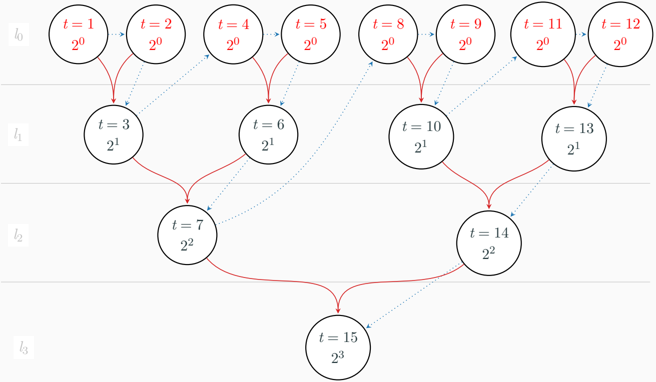


(b) Tree-structured view

From sequential Luby to arborescent Luby

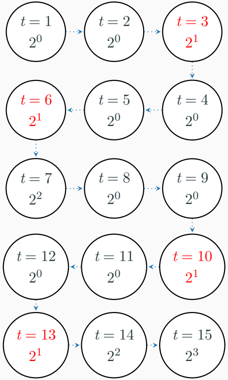


(a) Sequential view

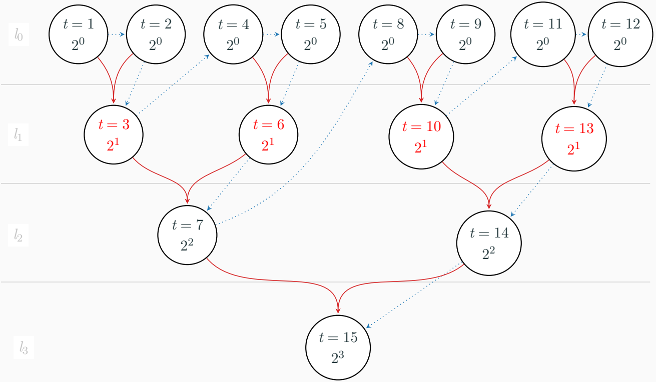


(b) Tree-structured view

From sequential Luby to arborescent Luby

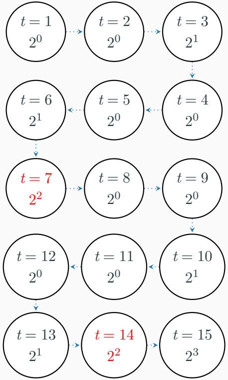


(a) Sequential view

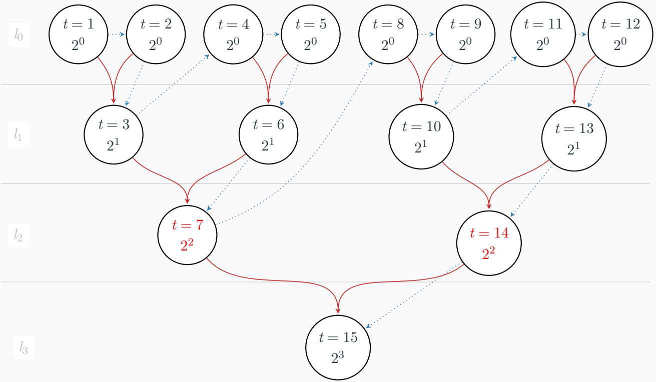


(b) Tree-structured view

From sequential Luby to arborescent Luby

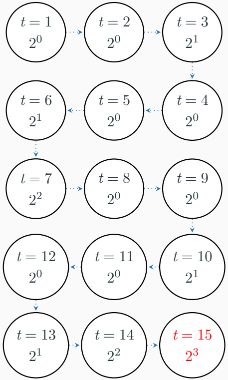


(a) Sequential view

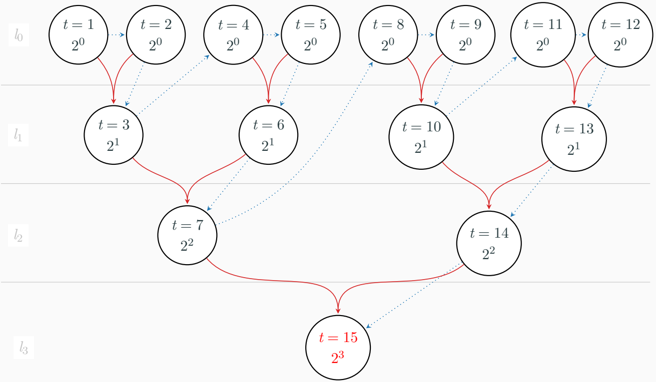


(b) Tree-structured view

From sequential Luby to arborescent Luby



(a) Sequential view



(b) Tree-structured view

AST bandit algorithm

Algorithm: Adaptive Single Tournament (AST)

Input: A set of arms $[K]$, a positive integer $m \geq 1$

```
Set S = [K]
for each run t = 1, 2, ... do
    if σ_luby(t) = 1 then
        Select an arbitrary arm i ∈ S
        Set S = S \ {i} and if S = ∅ then set S = [K]
    else
        Let i_left be the arm played at run t − σ_luby(t)
        Let i_right be the arm played at run t − 1
        Choose i ∈ {i_left, i_right} with best reward r_i
    Play i for m times and set r_i to the mth observed reward at run t
```

AST bandit algorithm

Algorithm: Adaptive Single Tournament (AST)

Input: A set of arms $[K]$, a positive integer $m \geq 1$

```
Set S = [K]
for each run t = 1, 2, ... do
    if σ_luby(t) = 1 then
        Select an arbitrary arm i ∈ S
        Set S = S \ {i} and if S = ∅ then set S = [K]
    else
        Let i_left be the arm played at run t − σ_luby(t)
        Let i_right be the arm played at run t − 1
        Choose i ∈ {i_left, i_right} with best reward r_i
    Play i for m times and set r_i to the mth observed reward at run t
```

AST bandit algorithm

Algorithm: Adaptive Single Tournament (AST)

Input: A set of arms $[K]$, a positive integer $m \geq 1$

```
Set S = [K]
for each run t = 1, 2, ... do
    if σ_luby(t) = 1 then
        Select an arbitrary arm i ∈ S
        Set S = S \ {i} and if S = ∅ then set S = [K]
    else
        Let i_left be the arm played at run t − σ_luby(t)
        Let i_right be the arm played at run t − 1
        Choose i ∈ {i_left, i_right} with best reward r_i
    Play i for m times and set r_i to the mth observed reward at run t
```

AST bandit algorithm

Algorithm: Adaptive Single Tournament (AST)

Input: A set of arms $[K]$, a positive integer $m \geq 1$

```
Set S = [K]
for each run t = 1, 2, ... do
    if σ_luby(t) = 1 then
        Select an arbitrary arm i ∈ S
        Set S = S \ {i} and if S = ∅ then set S = [K]
    else
        Let i_left be the arm played at run t − σ_luby(t)
        Let i_right be the arm played at run t − 1
        Choose i ∈ {i_left, i_right} with best reward r_i
    Play i for m times and set r_i to the mth observed reward at run t
```

AST bandit algorithm

Algorithm: Adaptive Single Tournament (AST)

Input: A set of arms $[K]$, a positive integer $m \geq 1$

Set $S = [K]$
for *each run* $t = 1, 2, \ldots$ **do**
    if $\sigma_{\text{luby}}(t) = 1$ **then**
        Select an arbitrary arm $i \in S$
        Set $S = S \setminus \{i\}$ and **if** $S = \emptyset$ **then** set $S = [K]$
    else
        Let i_{left} be the arm played at run $t - \sigma_{\text{luby}}(t)$
        Let i_{right} be the arm played at run $t - 1$
        Choose $i \in \{i_{\text{left}}, i_{\text{right}}\}$ with best reward r_i
    Play i for m times and set r_i to the mth observed reward at run t

Experiments

Experimental context

CSP instances ($\mathcal{I}_{\text{CSP}}$): 810 *instances* XCSP'$17/18/19$ (83 *families*)

Experimental context

CSP instances ($\mathcal{I}_{\text{CSP}}$):	810 *instances* XCSP'$17/18/19$ (83 *families*)
computation nodes:	3.3 *GHz* CPU Intel XEON E5-2643 and 32 *GB of RAM*

Experimental context

CSP instances ($\mathcal{I}_{\text{CSP}}$):	810 *instances* XCSP'$17/18/19$ (83 *families*)
computation nodes:	3.3 *GHz* CPU Intel XEON E5-2643 and 32 *GB of RAM*
timeout:	$2,400$ *seconds*

Experimental context

CSP instances ($\mathcal{I}_{\texttt{CSP}}$):	810 *instances* XCSP'$17/18/19$ (83 *families*)
computation nodes:	3.3 *GHz* CPU Intel XEON E5-2643 and 32 *GB of RAM*
timeout:	$2,400$ *seconds*
solvers:	`ACE` with each heuristics $\mathcal{H}$ and policies $\mathcal{B}$

Experimental context

CSP instances ($\mathcal{I}_{\texttt{CSP}}$):	810 *instances* XCSP'$17/18/19$ (83 *families*)
computation nodes:	3.3 *GHz* CPU Intel XEON E5-2643 and 32 *GB of RAM*
timeout:	$2,400$ *seconds*
solvers:	`ACE` with each heuristics $\mathcal{H}$ and policies $\mathcal{B}$
restarting sequence:	$u \times \texttt{luby}_t$ where $u = 150$

Experimental context

CSP instances ($\mathcal{I}_{\text{CSP}}$):	810 *instances* XCSP'$17/18/19$ (83 *families*)
computation nodes:	3.3 *GHz* CPU Intel XEON E5-2643 and 32 *GB of RAM*
timeout:	$2,400$ *seconds*
solvers:	`ACE` with each heuristics $\mathcal{H}$ and policies $\mathcal{B}$
restarting sequence:	$u \times \texttt{luby}_t$ where $u = 150$
cutoff unity:	*wrong decisions*

Experimental context

CSP instances ($\mathcal{I}_{\text{CSP}}$):	810 *instances* XCSP'$17/18/19$ (83 *families*)
computation nodes:	3.3 *GHz* CPU Intel XEON E5-2643 and 32 *GB of RAM*
timeout:	$2,400$ *seconds*
solvers:	`ACE` with each heuristics $\mathcal{H}$ and policies $\mathcal{B}$
restarting sequence:	$u \times \texttt{luby}_t$ where $u = 150$
cutoff unity:	*wrong decisions*
value ordering heuristic:	`min-dom`

Experimental context

CSP instances ($\mathcal{I}_{\text{CSP}}$):	810 *instances* XCSP'$17/18/19$ (83 *families*)
computation nodes:	3.3 *GHz* CPU Intel XEON E5-2643 and 32 *GB of RAM*
timeout:	$2,400$ *seconds*
solvers:	`ACE` with each heuristics $\mathcal{H}$ and policies $\mathcal{B}$
restarting sequence:	$u \times \texttt{luby}_t$ where $u = 150$
cutoff unity:	*wrong decisions*
value ordering heuristic:	`min-dom`
propagation property:	*arc-consistency*

Experimental context

CSP instances ($\mathcal{I}_{\text{CSP}}$):	810 *instances* XCSP'$17/18/19$ (83 *families*)
computation nodes:	3.3 *GHz* CPU Intel XEON E5-2643 and 32 *GB of RAM*
timeout:	$2,400$ *seconds*
solvers:	`ACE` with each heuristics $\mathcal{H}$ and policies $\mathcal{B}$
restarting sequence:	$u \times \texttt{luby}_t$ where $u = 150$
cutoff unity:	*wrong decisions*
value ordering heuristic:	`min-dom`
propagation property:	*arc-consistency*
learning:	*nogoods* at the end of runs

Overall analysis

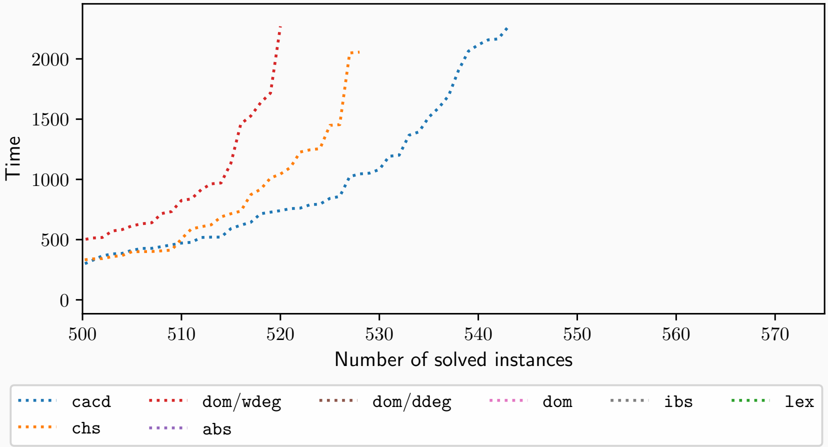


Figure 2: Cactus plots of the branching strategies

Overall analysis

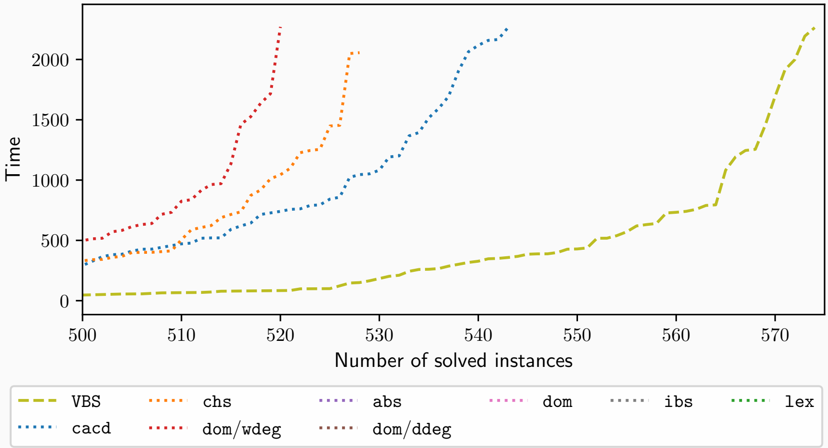


Figure 2: Cactus plots of the branching strategies

Overall analysis

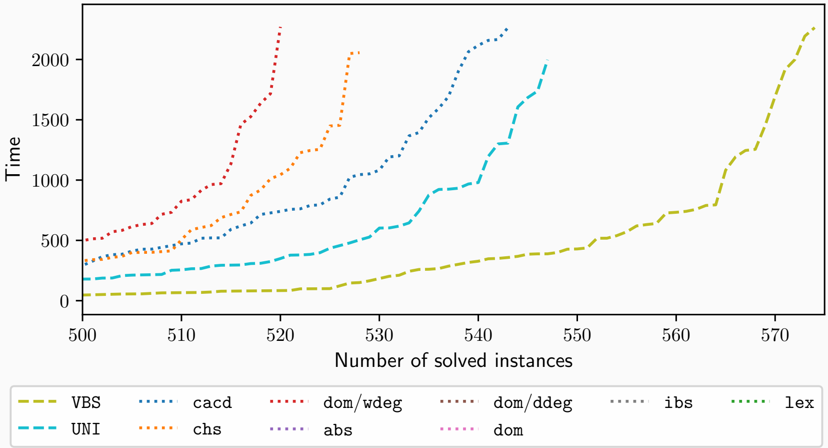


Figure 2: Cactus plots of the branching strategies

Overall analysis

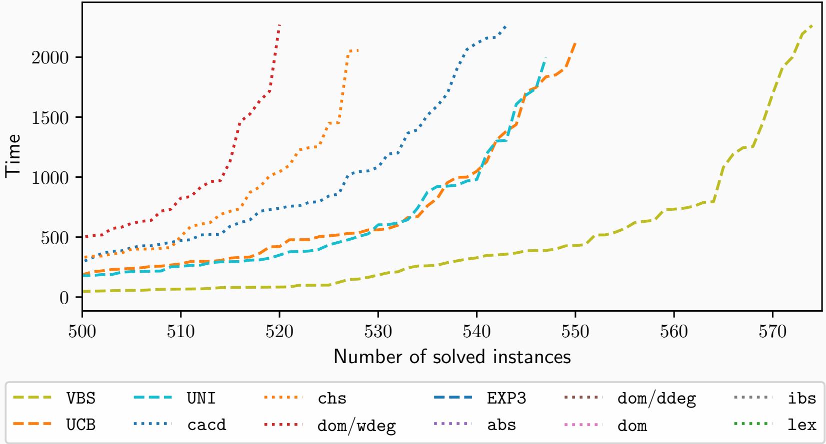


Figure 2: Cactus plots of the branching strategies

Overall analysis

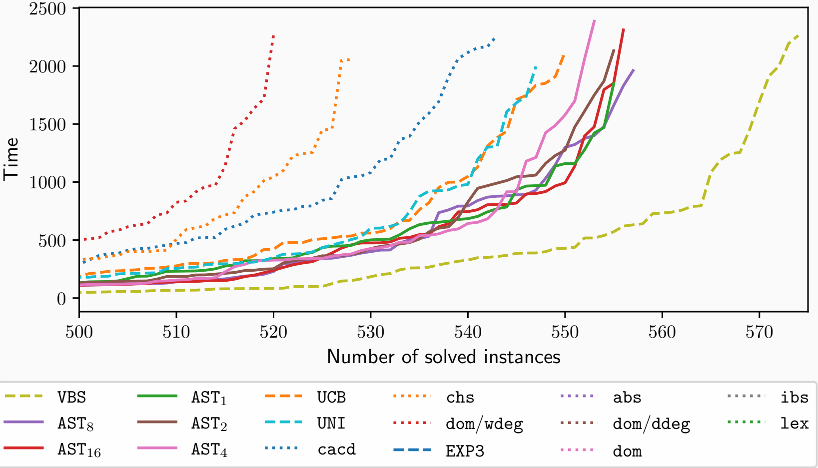


Figure 2: Cactus plots of the branching strategies

Finer-grained analysis

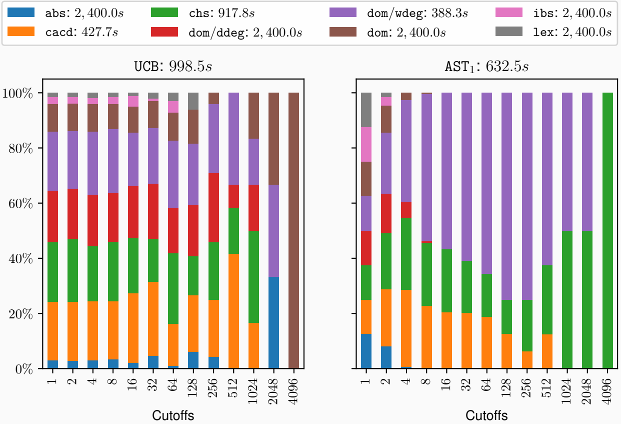


Figure 3: Proportions of heuristics selected by UCB and AST at each cutoff of Luby's sequence, for the CSP instance *Rlfap-scen-11-f01_c18*

Conclusion

Conclusion

In this study, we have:

- focused on the best heuristic identification problem
- presented the non-stochastic bandit algorithm AST

The results have shown a better behaviour than the stochastic and adversarial bandits-based, and are closer to the VBS. In addition, these results are corroborated by a convergence analysis.

Conclusion

In this study, we have:

- focused on the best heuristic identification problem
- presented the non-stochastic bandit algorithm AST

The results have shown a better behaviour than the stochastic and adversarial bandits-based, and are closer to the VBS. In addition, these results are corroborated by a convergence analysis.

Conclusion

In this study, we have:

- focused on the best heuristic identification problem
- presented the non-stochastic bandit algorithm AST

The results have shown a better behaviour than the stochastic and adversarial bandits-based, and are closer to the VBS. In addition, these results are corroborated by a convergence analysis.

Best Heuristic Identification for Constraint Satisfaction

Frederic Koriche, Christophe Lecoutre, Anastasia Paparrizou, **Hugues Wattez**

11 octobre 2022

Présentation à la 6ème journée CAVIAR

Perspective

As perspectives, we think:

- designing bandit algorithm for others universal restart schemes (*e.g.*, exponential sequence)
- extending learning and autonomy (*e.g.*, branching heuristics and propagation techniques)

Perspective

As perspectives, we think:

- designing bandit algorithm for others universal restart schemes (*e.g.*, exponential sequence)
- extending learning and autonomy (*e.g.*, branching heuristics and propagation techniques)