



SNT

Selecting Search Strategy in Constraint Solvers using Bayesian Optimization

Hedieh Haddad, Pierre Talbot, Pascal Bouvry

hedieh.haddad@uni.lu

October 30 - 2024

Background: Constraint Programming

- Constraint programming (CP) is a paradigm for solving combinatorial problems that can be applied to a wide range of problem domains.
- In CP, users declaratively state the constraints on a set of decision variables to find a feasible solution.
- CP solvers use a search strategy to explore the space of possible solutions.



A Challenge in CP

- Constraint programming solvers are black-box functions with many parameters.
- Efficiency of constraint programming solvers depends heavily on their parameters.
- A lot of possible parameters, but a set of parameters not always good on each problem (no-free-lunch theorem).
- It is left to the user to manually pick the best set of parameters to obtain the best efficiency.
 - significant impact on the efficiency of the solver

Hyperparameter Optimisation (HPO)

- HPO is the process of selecting the optimal values for an algorithm's hyperparameters.
- HPO is very successful in other fields like ML.
- HPO can improve tremendously the efficiency of the algorithm in ML.

Can hyperparameter optimisation improve the efficiency of constraint programming solvers?

HPO for CP

- Problem:
 - The numerous hyperparameters in CP solvers hinder the efficiency of HPO due to the large state-space.
- Solution:
 - Focus on particular and impactful subset of hyperparameters: search strategy.
 - We propose to encode the search strategy as a set of hyperparameters optimised using a HPO algorithms.

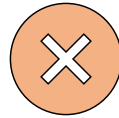
Why Search Strategies Matter?

The Role in Constraint Programming



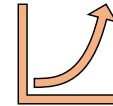
Core of Solver Efficiency

Search strategies determine how a solver navigates the solution space, directly impacting performance.



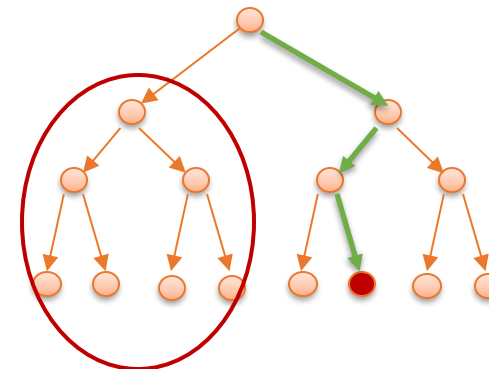
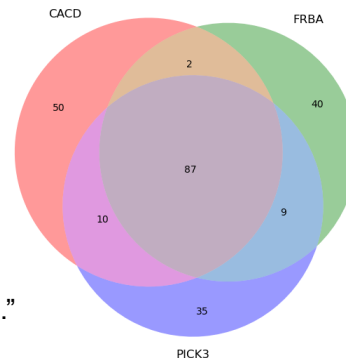
No Universal Strategy

No single strategy works best for all problems.




Need for Optimization

Optimizing search strategies per problem is essential for maximizing solver efficiency and effectiveness.



Hyperparameter Optimisation (HPO)

There are several algorithms for hyperparameter optimisation, including:

- Grid search
- Random search
- Hyper-band optimisation
- **Bayesian optimisation** 
- ...

Bayesian Optimisation is approved as the most effective HPO methods between the specified HPO methods that we have tried during the benchmarking. ¹

1. H. Haddad, P. Talbot, and P. Bouvry, "Comparison of Hyperparameter Optimization Methods for Selecting Search Strategy of Constraint Programming Solvers," 7th Workshop on Progress Towards the Holy Grail (PTHG-24), A CP 2024 Workshop, September 2024.

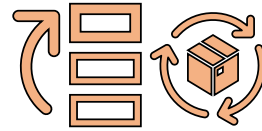
Probe and Solve Algorithm

Two-Phase Approach for Optimizing Search Strategies



Probing Phase

Explores various search strategies using HPO methods, ranking them based on performance within a limited time frame, defined as **K** percent of the total available time.



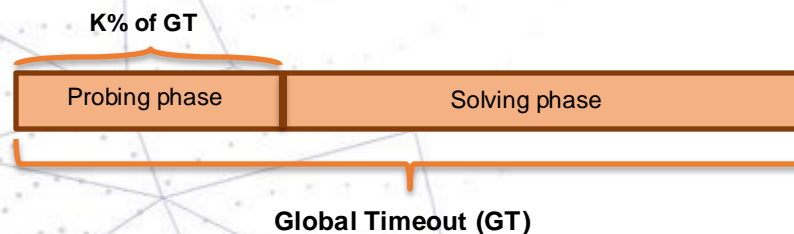
Solving Phase

Utilizes the top-ranked strategy from the probing phase to solve the constraint problem.



Dynamic Timeout Adjustment

Algorithm adapts dynamically based on problem complexity and solver performance, enhancing efficiency.



Probe and Solve Algorithm

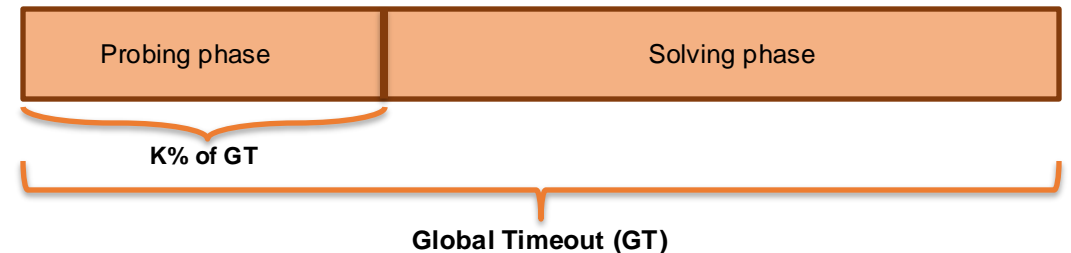
Two-phase algorithm:

1. Probing phase

- ✓ Constant **K** percent of global time
- ✓ Using the HPO methods to rank the search strategies

2. Solving phase

- ✓ Solving the problem with the best configuration



- ❖ **General algorithm - can be used with any constraint solvers**
- ❖ **Completely implemented in Python**
- ❖ **Compatible with 2 popular frameworks (Minizinc/XCSP3)**

Probe and Solve Algorithm

➤ Pseudo-Code :

```

function PSA( $\langle X, D, C, obj \rangle$ ,  $hpo$ ,  $HP$ ,  $GT$ ,  $PT$ )
  Initialize  $ET$  to 0 seconds
  Initialize  $best\_obj$  to  $\infty$ 
  while  $ET < PT$  do
     $psolve \leftarrow \lambda s.solve(\langle X, D, C, obj \rangle, s, CT)$ 
     $ranking, obj \leftarrow hpo(HP, psolve)$ 
     $ET \leftarrow ET + CT$ 
    if  $obj \neq \infty$  then
       $min(obj, best\_obj)$ 
    else
       $CT \leftarrow CT \times Geometric\_Coefficient$ 
    end if
  end while
  if  $best\_obj = \infty$  then
    return  $solve(\langle X, D, C, obj \rangle, ranking[0], GT - PT)$ 
  else
    return  $min(best\_obj, solve(\langle X, D, C \wedge obj < best\_obj, obj \rangle, ranking[0], GT - PT))$ 
  end if
end function

```

Validating the Robustness with Shorter Timeouts

To determine if shorter timeouts can achieve promising results comparable to longer timeouts by examining the correlation of search strategy rankings at both shorter probing timeout and global timeouts:

- 2 statistical methods Used: Spearman's Rank Correlation, Kendall's Tau Correlation
- 4 randomly chosen problems
- 4 different timeouts: 5% , 10%, 20%, 50%

| Instance | 5 | | 10 | | 20 | | 50 | |
|------------------------------|----------|-------------|----------|-------------|----------|-------------|----------|-------------|
| | Spearman | Kendall Tau | Spearman | Kendall Tau | Spearman | Kendall Tau | Spearman | Kendall Tau |
| CarpetCutting-test05 | 0.94 | 0.83 | 0.96 | 0.92 | 0.91 | 0.84 | 0.89 | 0.81 |
| GeneralizedMKP-OR05x100-75-1 | 0.99 | 0.99 | 0.99 | 0.99 | 0.92 | 0.84 | 0.91 | 0.80 |
| RIP-25-0-j120-01-01 | -0.33 | -0.33 | 0.88 | 0.79 | 0.92 | 0.82 | 0.97 | 0.89 |
| KidneyExchange-4-081 | 0.83 | 0.83 | 0.87 | 0.84 | 0.90 | 0.82 | 0.93 | 0.82 |

* High correlation suggests that rankings from shorter timeouts are like those from longer timeouts.

An Extended Study

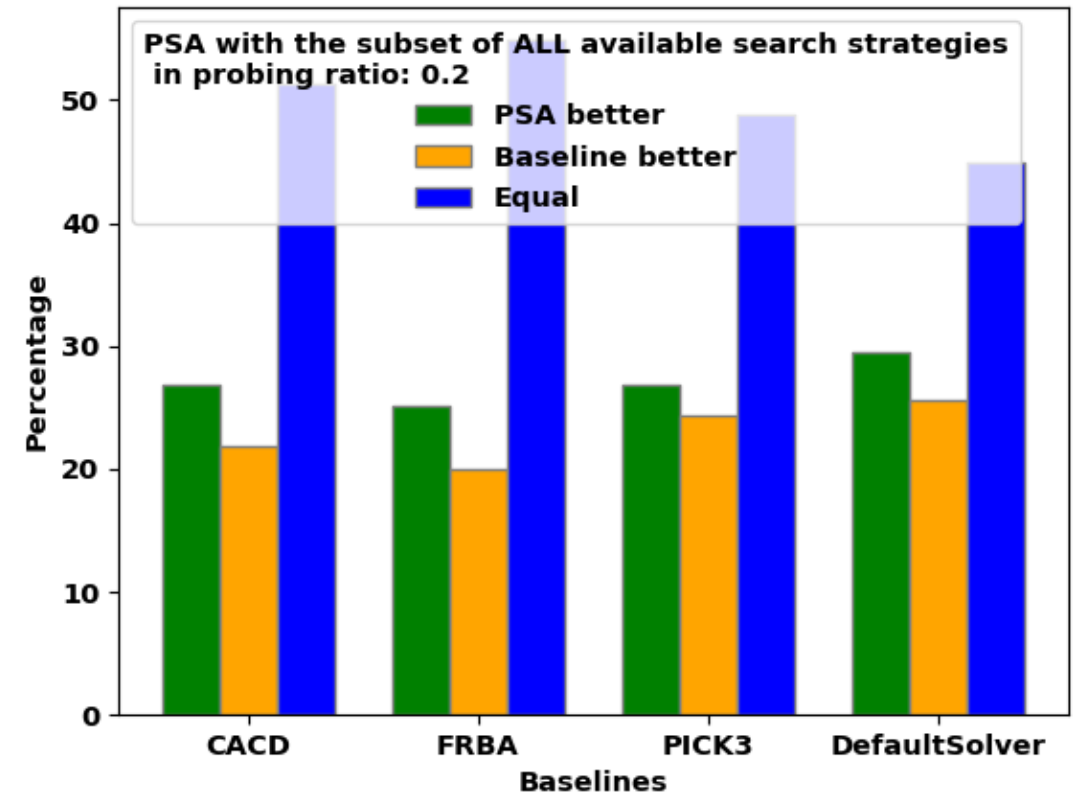
Study other questions using PSA

- PSA can be used to analyse the efficiency of various subset of search strategies which are frequently studied within constraint programming:
 - Do dynamic search strategies outperform static ones?
 - Does assigning different strategies to different subsets of the problem with different objectives, lead to better results?
 - Can tuning more solver parameters, improve performance?

Experiment Results (*XCSP3 Benchmark with ACE Solver*)

ALL available search strategies provided by the solver:

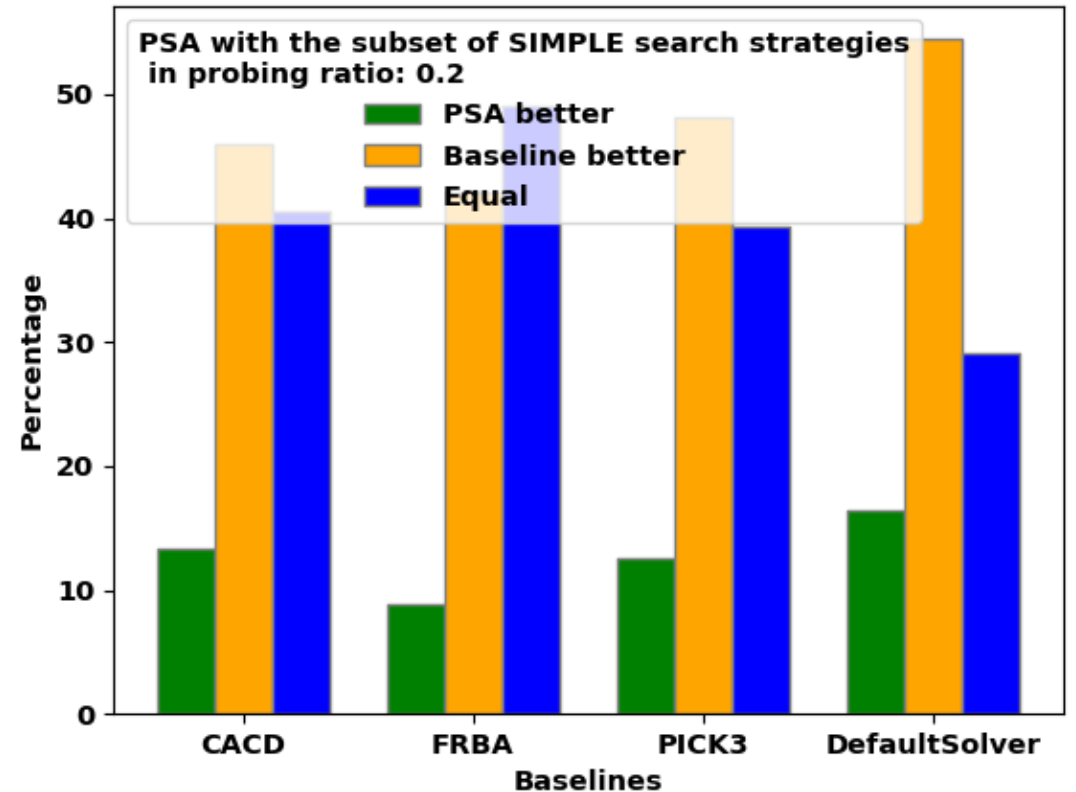
- Explore all search strategies offered by the ACE solver.
- Compared to the default and three well-known universal search strategies (CACD, FRBA, PICK3).
- Aim to do a bit better than all individual search strategies, as finding a good universal search strategy is very hard due to the variability in problem instances.
- **PSA could outperform the baselines, around 28%-30% of the cases.**



Experiment Results (*XCSP3 Benchmark with ACE Solver*)

STATIC search strategies provided by the solver:

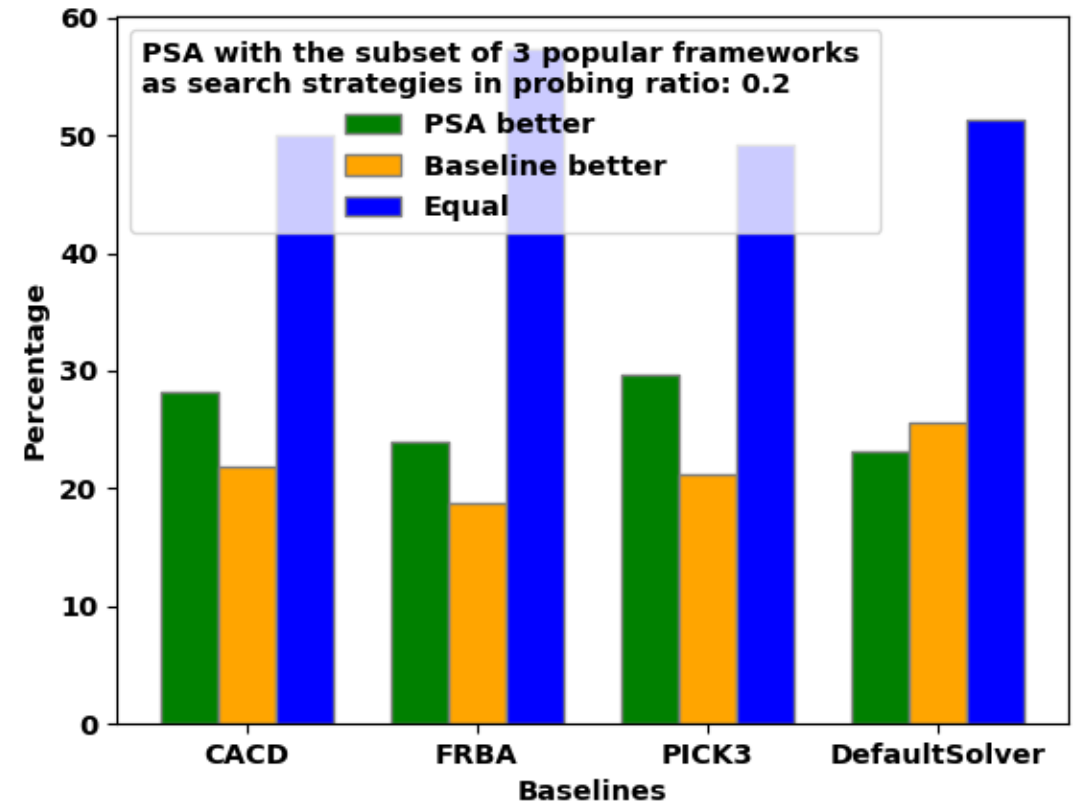
- Search strategies that are simpler to implement and do not require collecting statistics of propagators.
- In order to avoid calculation and update weights during the solving process.
- The baselines (CACD, FRBA, PICK3) are dynamic variable selection strategies.
- **The baseline methods demonstrate superior performance around 50% of the cases.**



Experiment Results (*XCSP3 Benchmark with ACE Solver*)

3 DYNAMIC variable selection strategy:

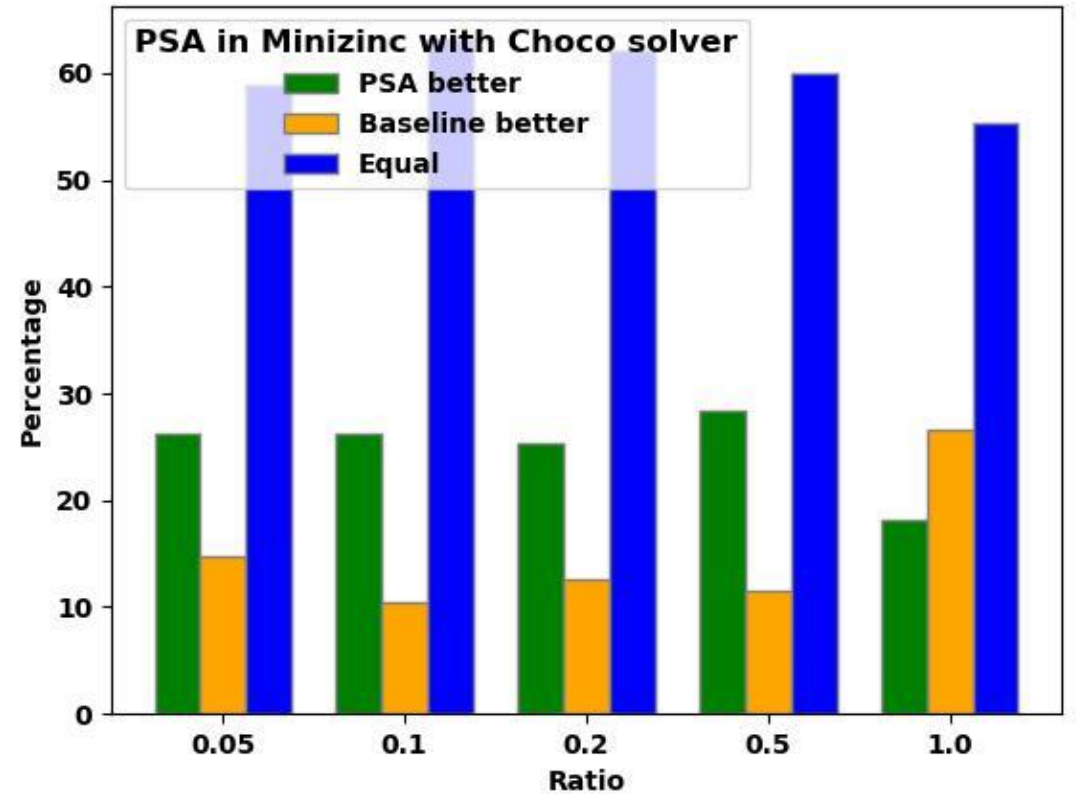
- Showing the effectiveness of the three well-known universal search strategies: PICK3, CACD, and FRBA
- **PSA outperforms the baselines search strategy, around 30% of the cases**



Experiment Results (*Minizinc Benchmark with Choco Solver*)

ALL available search strategies provided by the solver:

- Explore all search strategies offered by the Choco solver.
- Compare to the default search strategy provided by the solver which is the combination of (*DomWDeg, Indomain_Min*)
- **With the ratio of 0.5, PSA outperforms the default search strategy in 28.42% of the instances.**



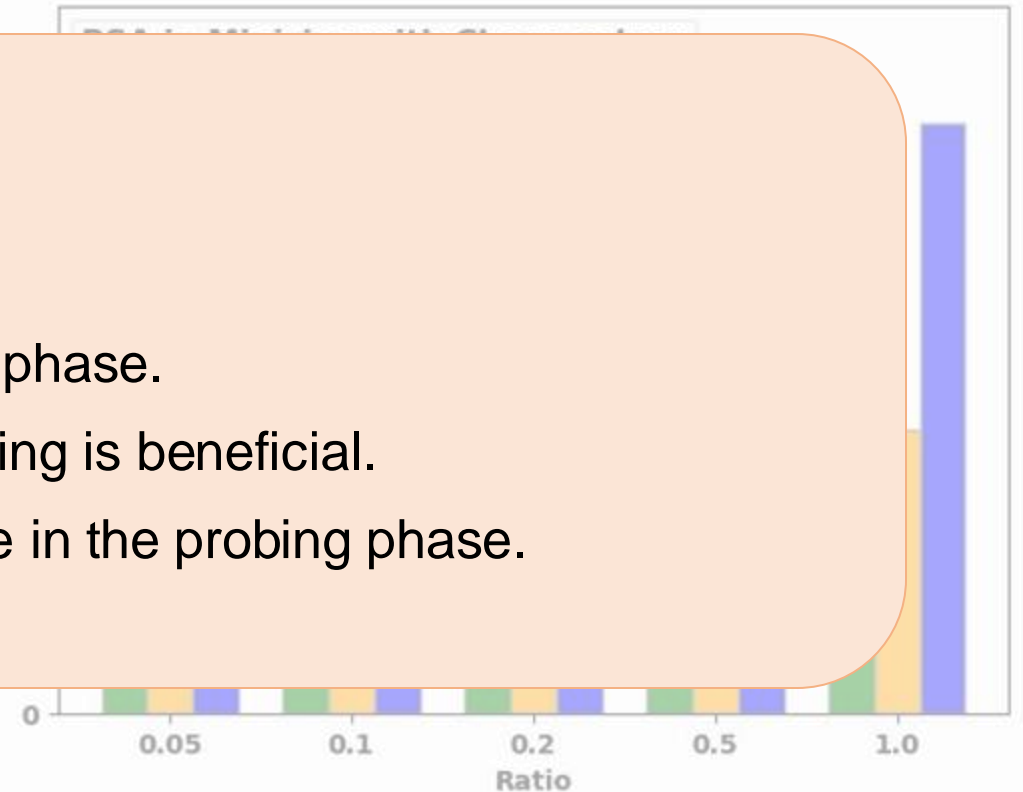
Experiment Results (*Minizinc Benchmark with Choco Solver*)

ALL a

- Ex
- Co
- wh
- Wi
- stra

Observation:

- Results indicate the importance of the probing phase.
- Spending a portion of time (e.g., 50%) on probing is beneficial.
- Demonstrates the advantages of investing time in the probing phase.



Conclusion

PSA validation

- We designed a new algorithm that applies HPO methods to CP solvers called PSA.
- We focused on the most important hyperparameters of a constraint solvers named search strategies.
- PSA outperformed baselines using Bayesian optimization around 30% of the cases.
- PSA is a generic and parameter-less approach that can be used on top of any MiniZinc-compatible or XCSP3-compatible solvers, without modifying those.

Thank you



hedieh.haddad@uni.lu