Comparison of Hyperparameter Optimization Methods for Selecting Search Strategy of Constraint Programming Solvers

Hedieh Haddad, Pierre Talbot, Pascal Bouvry

Hedieh.haddad@uni.lu

2nd September

Background: 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 the 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?

- ***If yes, which HPO method works better?***

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 the HPO algorithms.
- **WHY?**

Why Search Strategies Matter?

The Role in Constraint Programming

Core of Solver Efficiency

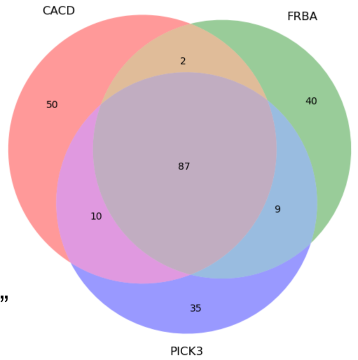
Search strategies determine how a solver navigates the solution space, directly impacting performance.

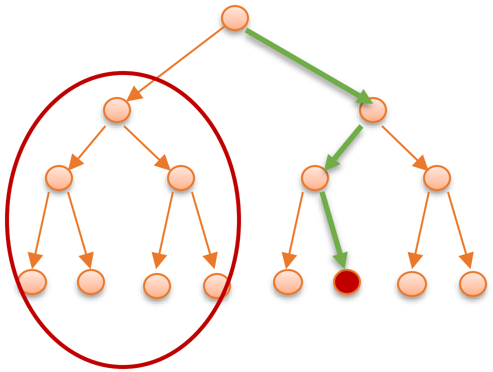
No Universal Strategy

No single strategy works best for all problems.

Need for Optimization

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





“Guiding Backtrack Search by Tracking Variables During Constraint Propagation.”
G. Audemard, C. Lecoutre, and C. Prud’homme

Hyperparameter Optimisation (HPO)

There are several algorithms for hyperparameter optimisation, including:

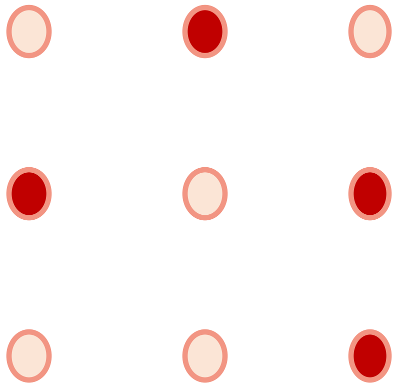
- Grid search
- Random search
- Hyper-band optimisation
- Bayesian optimization
- …

Hyperparameter Optimisation (HPO)

There are several algorithms for hyper-parameter optimisation, including:

- Grid search
- Random search
- Hyper-band optimisation
- Bayesian optimization
- …

selects random combinations of hyperparameters to evaluate

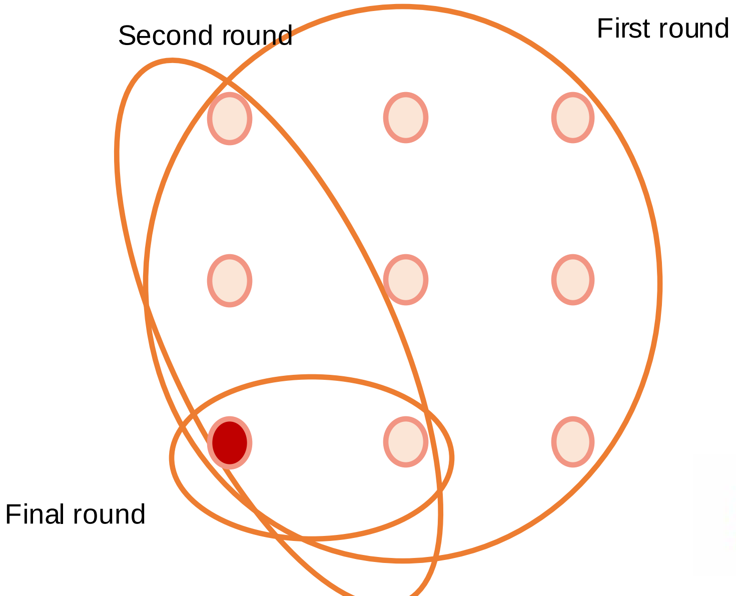


Hyperparameter Optimisation (HPO)

There are several algorithms for hyper-par

- Grid search
- Random search
- Hyper-band optimisation
- Bayesian optimization
- …

intelligently allocates resources to different configurations based on their performance

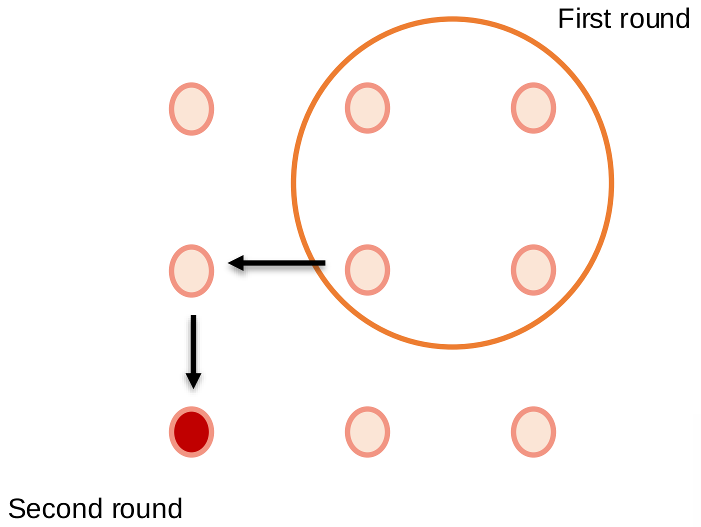


Hyperparameter Optimisation (HPO)

There are several algorithms for hyper-para

- Grid search
- Random search
- Hyper-band optimisation
- Bayesian optimization
- …

uses past evaluation results to choose the next set of hyperparameters to evaluate



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 (K percent) limited time.

Solving Phase

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

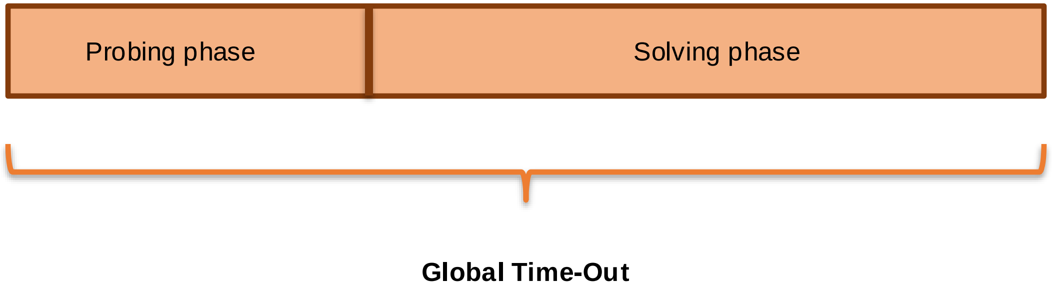
Flexibility

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**
- ❖ **Integrated in 2 popular frameworks (Minizinc/XCSP3)**

Implementation

- **Pseudo-Code :**

function $\text{PSA}(\langle X, D, C, obj\rangle, hpo, HP, GT$
  Initialize ET to 0 seconds
  Initialize $best\_obj$ to ∞
  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

Implementation

- **Github :**

  - https://github.com/Hedieh-Haddad/PSA.git

- **JSON :**

  - Choco solver in both frameworks

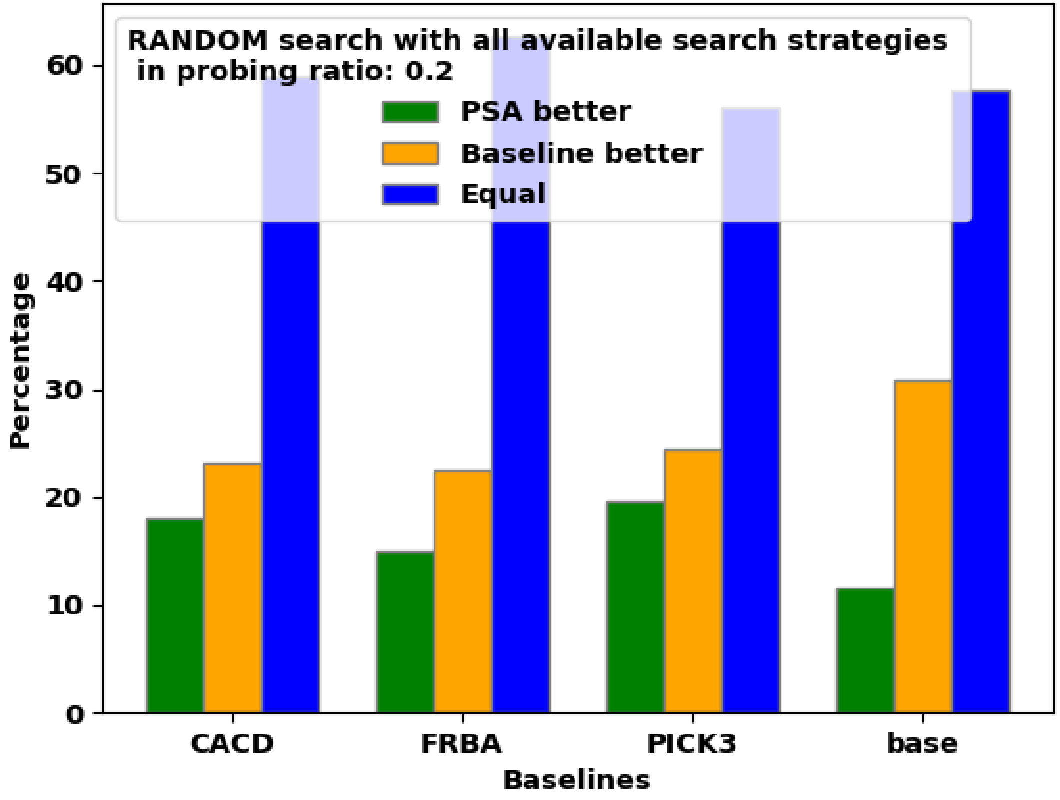

```
"Minizinc": {
    "search": {
        "varh_values": ["input_order", "first_fail", "smallest", "largest", "dom_w_deg", "occurrence", "most_constrained", "max_regret"],
        "valh_values": ["indomain_min", "indomain_max", "indomain_median", "indomain_random", "indomain_split", "indomain_reverse_split", "indomain_interval"],
        "default_varh": "dom_w_deg",
        "default_valh": "indomain_min"
    }},
"XCSP3": {
    "search": {
        "varh_values": ["DOM", "CHS", "DOMWDEG", "FLBA", "FRBA", "ACTIVITY" ,"INPUT", "RAND", "DOMWDEG_CACD", "FIRST_FAIL"],
        "valh_values": ["MAX", "MIN", "MED", "MIDFLOOR", "MIDCEIL", "RAND"],
        "default_varh": "DOMWDEG",
        "default_valh": "MIN"
    }
}
```

Experiment Results

Comparative Analysis of HPO Methods

Random Search

Showed variable results; not as robust as other methods due to inherent randomness.

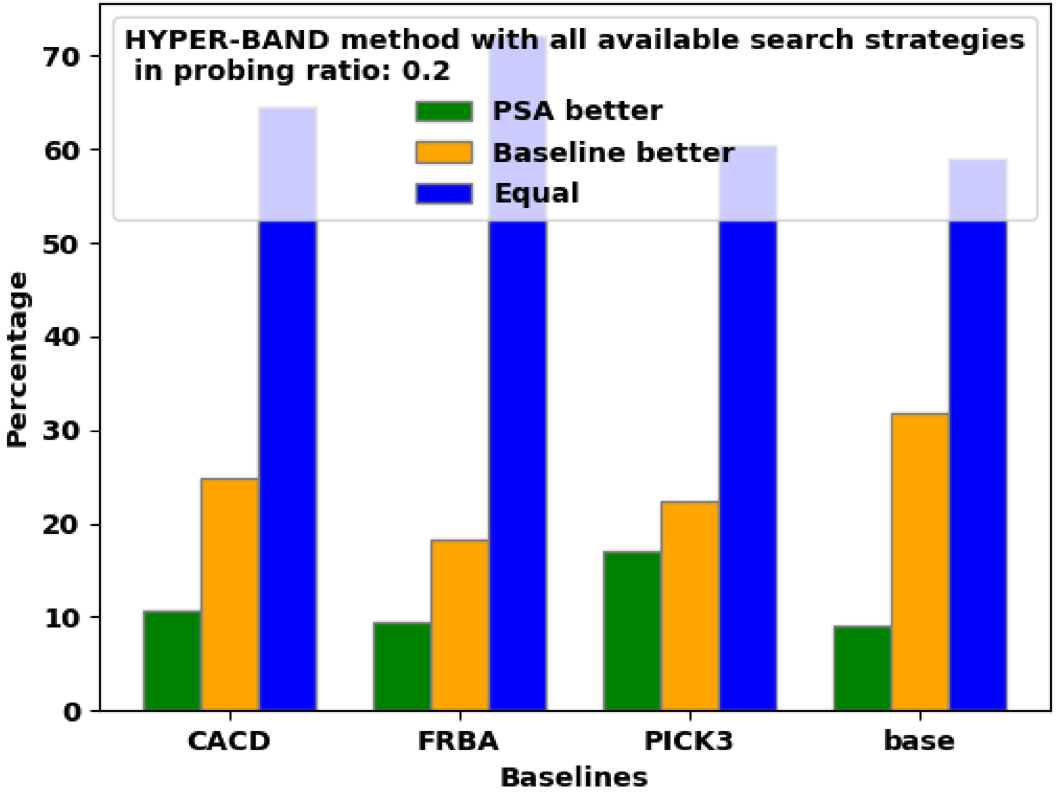


Experiment Results

Comparative Analysis of HPO Methods

Hyper-band Search

More efficient than random search in technique, but with significant variability in performance across different strategies.

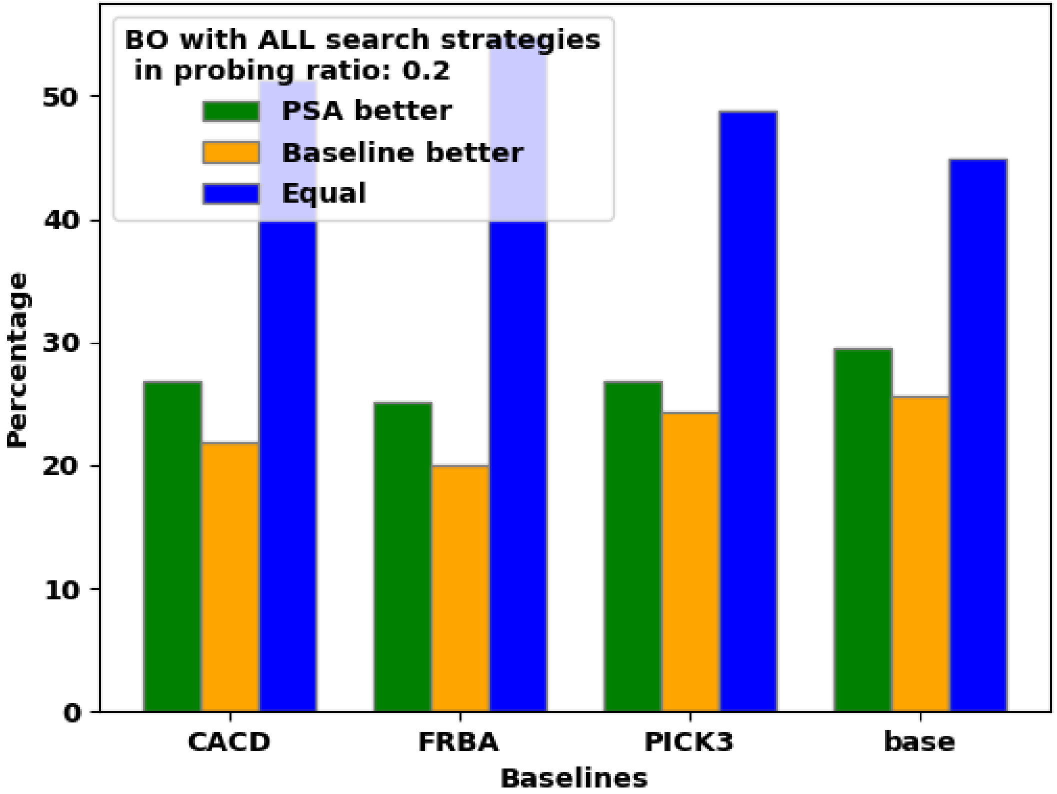


Experiment Results

Comparative Analysis of HPO Methods

Bayesian Optimization

Proved most effective, outperforming baseline strategies in around 30% of cases.



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 subsets lead to better results?
- Can tuning more solver parameters, improve performance?

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 generic and parameter less approach that can be used on top of any MiniZinc or XCSP3-compatible solvers, without modifying those.

Thank you

Questions?
Advice?

Hedieh.Haddad@uni.lu

Is the probing phase finding the best search strategy?

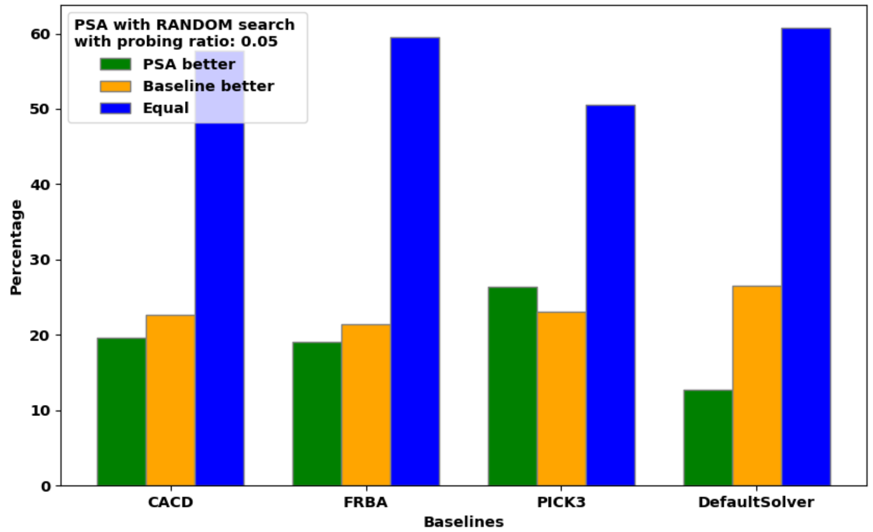
- 4 problems
- 4 different time-outs
  - %5 , %10, %20, %50
- All configuration
- Make ranking for each
- Comparison with %100

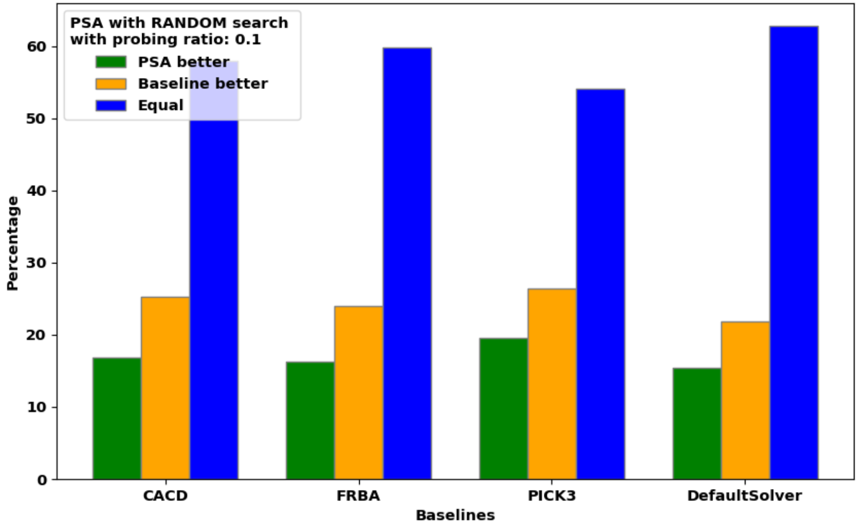
✓ Spearman's rank correlation

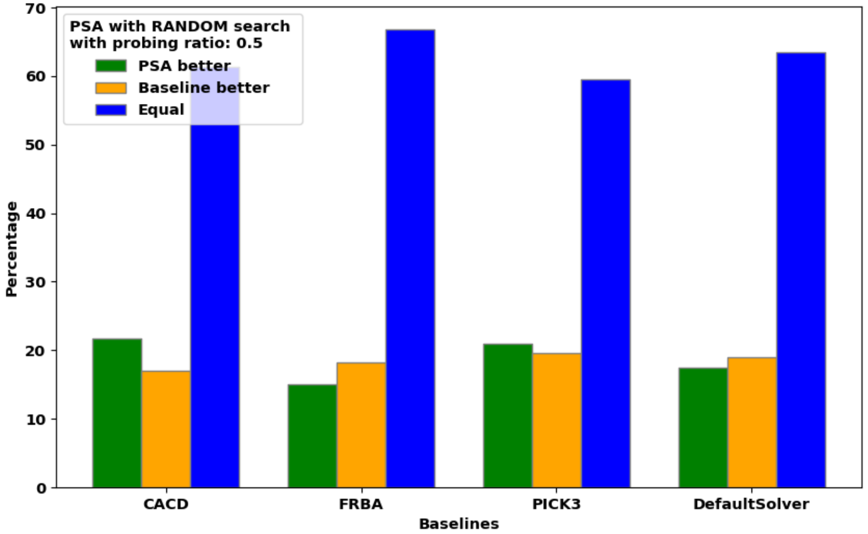
✓ Kendall's tau

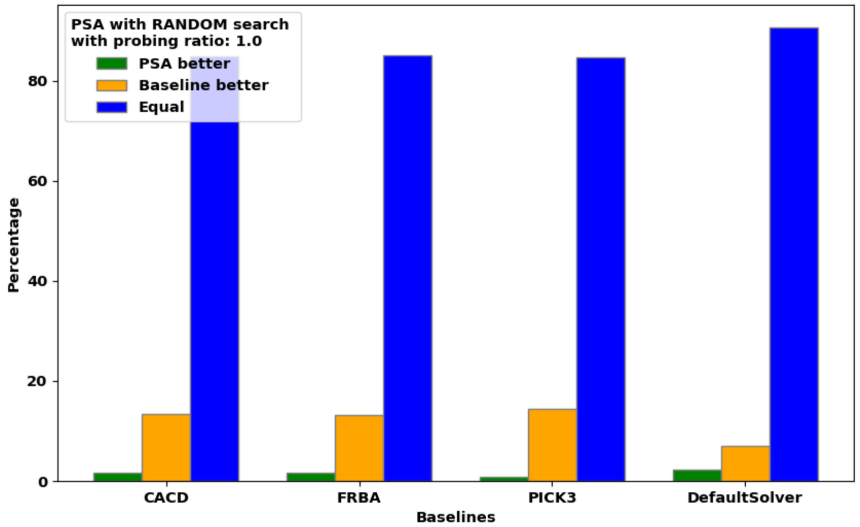
Instance	5		10		20		50	
	Spearman	Kendall Tau	Spearman	Kendall Tau	Spearman	Kendall Tau	Spearman	Kendall Tau
CarpetCutting-test05	94	83	96	92	91	84	89	81
GeneralizedMKP-OR05x100-75-1	99	99	99	99	92	84	91	80
RIP-25-0-j120-01-01	-33	-33	88	79	92	82	97	89
KidneyExchange-4-081	83	83	87	84	90	82	93	82

Experiment Results (Random Search)

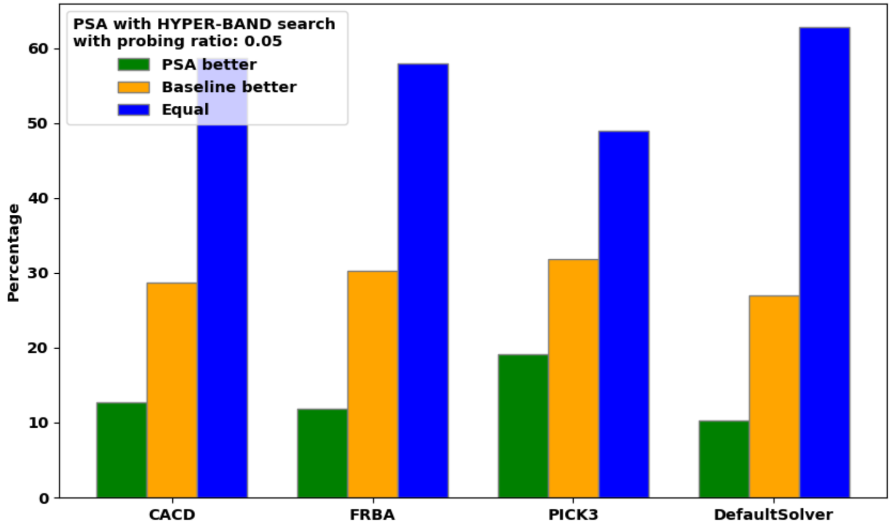


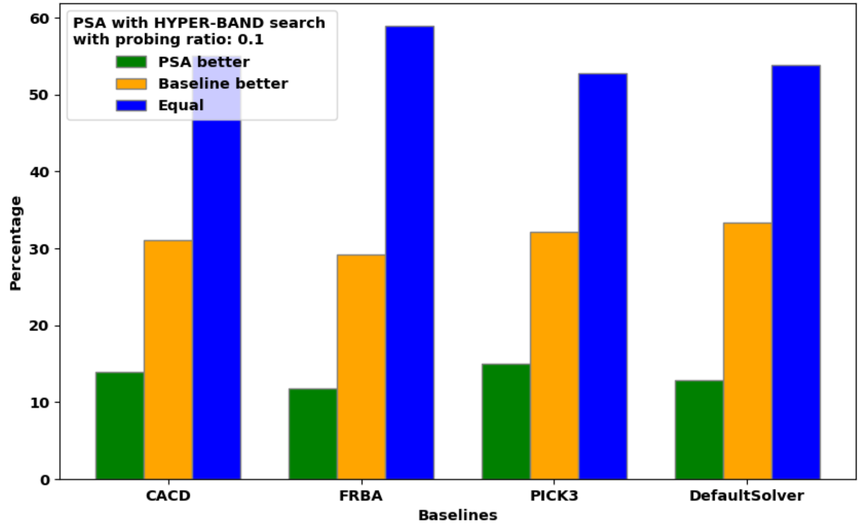


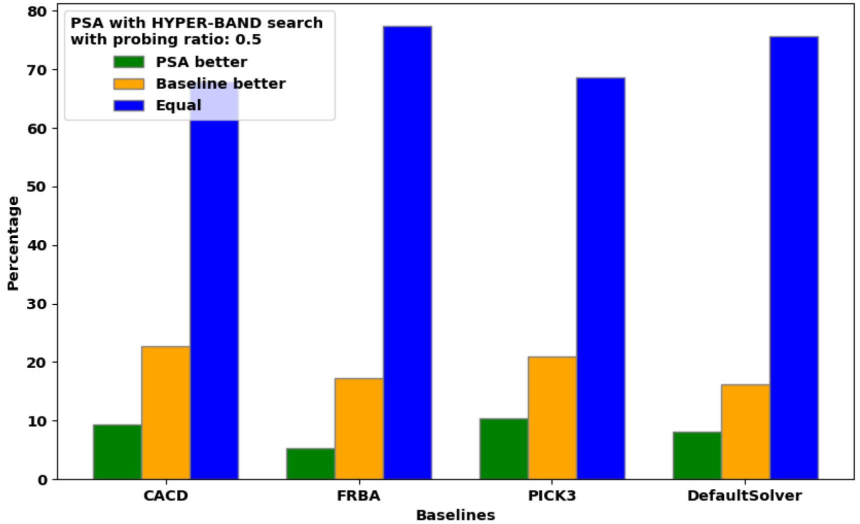


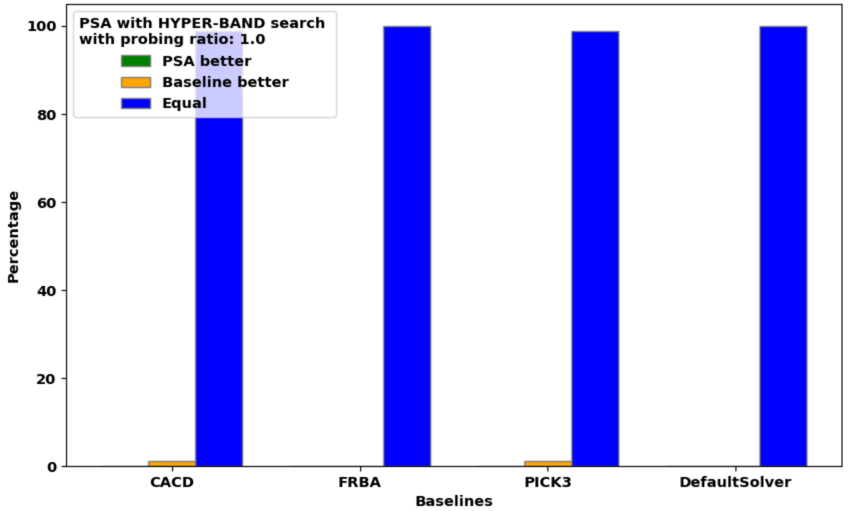


Experiment Results (Hyper-Band Search)

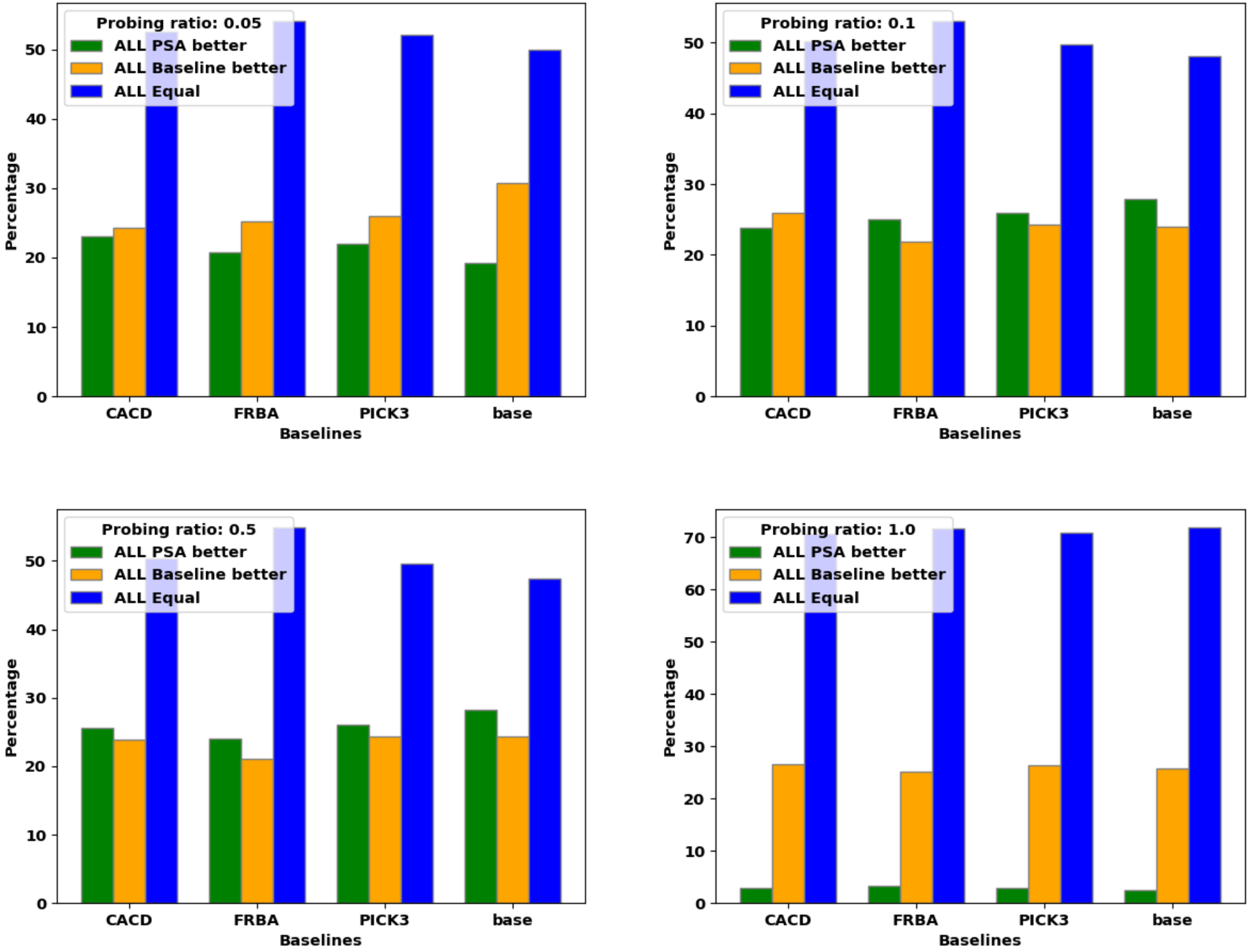








Experiment Results (Bayesian Optimisation Search)



Performance Across XCSP3 Benchmark

Comparison of Variable Selection Strategies

- **PickOnDom, FrbaOnDom, DomWDeg/CACD**

  - Three popular variable selection strategies widely used in constraint programming.

- **XCSP3 Competition 2023**

  - Comparison showed that none of these strategies consistently outperformed the others.

- **Overlap in Performance**

  - Objective values often matched across strategies, highlighting the need for tailored approaches.