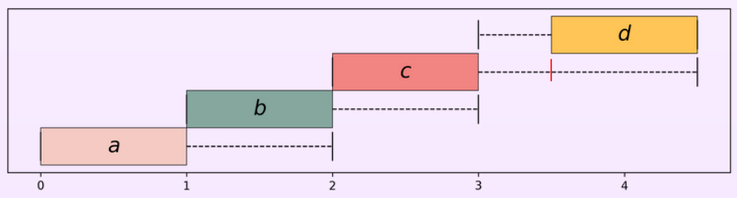
More general explanations for the Not-First/Not-Last propagators | Propagators for Constraint Programming

Yousef El Bakri (y.elbakri-1@student.tudelft.nl) | Supervised by: Emir Demirović, Imko Marijnissen

1 Background

Constraint Programming: Paradigm to solve NP-Hard problems with constraints and variables.
Lazy Clause Generation: Solvers that use explanations for better backtracking.
Disjunctive constraint: Forces tasks not to overlap.
Propagators: Functions that prune domains variables, and return an explanation.
Not-First/Not-Last propagators: Shorten the range of the tasks based on whether they can not be first or last compared to a subset of all tasks.
Naive explanations: Input the current state of all tasks.
Vilim's explanations [1]: Input only tasks that affect propagation, with an extended range which will still cause a conflict.
How to propagate with Not-Last [2]:

1. Generate set of tasks that could propagate a task
2. Check if the earliest completion time of the set is after the latest start time of the task
3. Change latest completion time of the task
4. Generate explanation using set of tasks



2 Research Questions

1. Can we create more general explanations by iterating through all subsets of tasks?
2. What are the effects of allowing the iteration method to a max size of k sized sets?

3 Experiments

Attempt to find a better subset only when tasks is at most length k.

Tested implementations:

- Naive explanations
- Vilim's explanations
- Values of k from 2 to 5 with Vilim's explanations

Data collected:

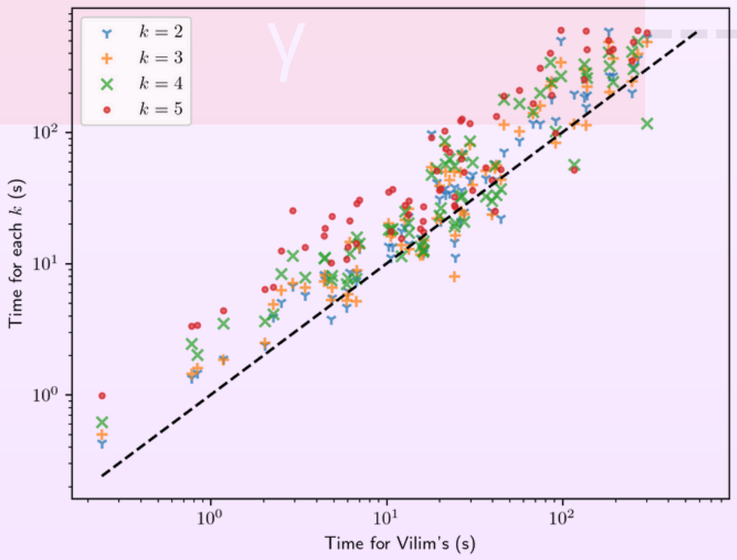
1. Time taken
2. Amount of conflicts
3. Average LBD [3]

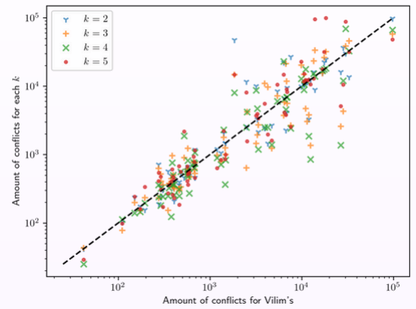
Tested against 70 instances:

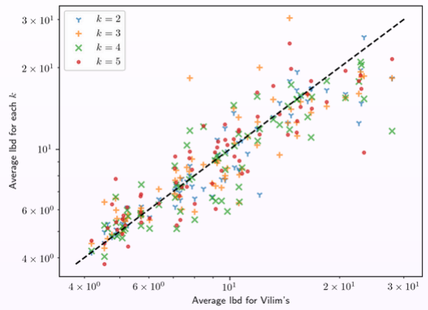
- LA01-LA40
- ORB01-ORB10
- 20 Generated TA instances

4 Results

- Almost all test instances are slower than Vilim's explanations.
- 50% less conflicts and lower average LBD.
- k = 4 had the best results compared to other values.
- All test instances were faster than naive explanations.







5 Discussion & Conclusion

Implementation of finding optimal subset had performance issues:

- Runtime was significantly affected by logging how often subsets were used.
- Implementation could be made better to use less memory, thus increasing speed.

Less conflicts found indicate a potential future in further exploring this field by:

- Testing higher values of k.
- Finding subsets faster with the use of heuristics.

We conclude:

- Looking through subsets of tasks does show an improvement in amount of conflicts and average LDB.
- Even with the increased complexity it still performed better than naive explanations.

References

[1] Computing Explanations for the Unary Resource Constraint. In Lecture Notes in Computer Science, pages 396-409. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
[2] Petr Vilím. Global constraints in scheduling. 2007. Publisher: Univerzita Karlova, Matematicko-fyzikální fakulta.
[3] Laurent Simon and Gilles Audemard. Predicting Learnt Clauses Quality in Modern SAT Solver. In Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09), Pasadena, United States, July 2009