Learning User Preferences in Interactive Constraint Programming

Mohamed Siala

LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France

February 20, 2024

Context & Related Work

Tough Decision, Nevertheless I Have a Taste

https://www.flickr.com/photos/160866001@N07/49857560608

- The variables are well defined
- The constraints are given
- The objective function is unknown: The user is non-expert in optimisation, aesthetic objective functions, dynamic environment, ..
- The user is able to rank the solutions according to her preferences
- Due to the exponential number of solutions, only a subset of solutions is iteratively proposed to the user

- A new research area ?

- A new research area ?
- It dates back to 1988 (before?) with the notion of dynamic CSPs Dechter and Dechter [1988]
- A lot of developments since then and in particular in the past decade due to the proliferation of its applications in the real word
- Modern prescriptive decision making relies heavily on data, feedback loops, machine learning, and flexible solvers
- Different types of interactions:
  - Problem definition
  - Parameters approximation
  - Evolution of the model
  - Emerging Patterns
  - Explanations
  - ...

Related Work (Non Exhaustive)

- Dynamic Constraint-Networks (Dechter and Dechter [1988])
- The Inductive Constraint Programming Loop (Bessiere et al. [2016])
- Predict+optimise (Demirovic et al. [2019])
- Constraint acquisition (Bessiere et al. [2017])
- Specific classes of objective functions Toffano et al. [2022]; Benabbou and Lust [2019]
- . . .

Our Framework

Problem Definition

- Initially, a set of solutions S is sampled

Problem Definition

- Initially, a set of solutions S is sampled
- A very limited number of interactions with the user is allowed

- Initially, a set of solutions S is sampled
- A very limited number of interactions with the user is allowed
- At each iteration, the user is given a subset of solutions of S to rank

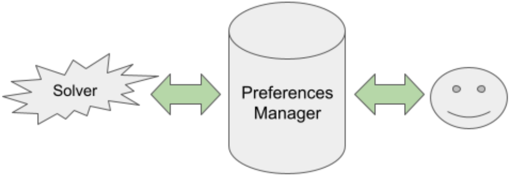
Problem Definition

- Initially, a set of solutions S is sampled
- A very limited number of interactions with the user is allowed
- At each iteration, the user is given a subset of solutions of S to rank
- In the first iteration, two random solutions are given to the user to compare

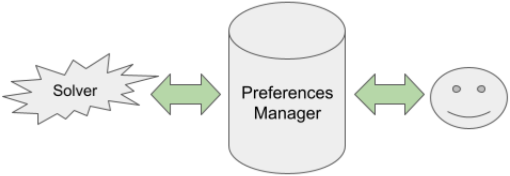
- Initially, a set of solutions S is sampled
- A very limited number of interactions with the user is allowed
- At each iteration, the user is given a subset of solutions of S to rank
- In the first iteration, two random solutions are given to the user to compare
- Then at each iteration i, one additional solution is proposed. The user updates its ranking accordingly

Problem Definition

- Initially, a set of solutions S is sampled
- A very limited number of interactions with the user is allowed
- At each iteration, the user is given a subset of solutions of S to rank
- In the first iteration, two random solutions are given to the user to compare
- Then at each iteration i, one additional solution is proposed. The user updates its ranking accordingly
- The purpose is to find the most preferred solutions in S within a bounded number of interactions



- Let S_i be the set of solutions that are ranked at iteration i
- The preferences manager build a ML model that predicts the ranking of the solutions in S by using S_i as a training data
- A solution with the best predicted rank is then proposed to the user at iteration $i+1$



- Let S_i be the set of solutions that are ranked at iteration i
- The preferences manager build a ML model that predicts the ranking of the solutions in S by using S_i as a training data
- A solution with the best predicted rank is then proposed to the user at iteration $i + 1$

There is a Problem!

- Let $[1..k]$ be the labels associated to the solutions in S_i
- A solution that is better than all the solutions in S_i must have a label $k + 1$.
- **How does one build a model that predicts a label that is not used in the training ?**

Straightforward Approach

- One can build a model for each label $l \in [1..k]$ to predict whether a given solution has label l
- **Weakness: This approach can be useful to predict solutions with different labels than $[1..k]$. However, it does not capture the notion of preferences**

Straightforward Approach

- One can build a model for each label $l \in [1..k]$ to predict whether a given solution has label l
- **Weakness: This approach can be useful to predict solutions with different labels than $[1..k]$. However, it does not capture the notion of preferences**

Proposed Approach

- By reasoning about the order between the solutions instead of the ranking, the prediction model learns what makes a solution better than another
- We propose to build a prediction model (denoted by O) that takes as input a couple (s_1, s_2) and outputs 1 is s_1 is better than s_2, -1 if s_1 is worse than s_2, and 0 if they have the same rank

- At each iteration, the training dataset is the set of all possible couples of solutions in S_i
- The labelling is inferred by the preferences of the user

- At each iteration, the training dataset is the set of all possible couples of solutions in S_i
- The labelling is inferred by the preferences of the user
- Once O is built, one might use different strategies to pick a new solution

The Proposed Approach

- At each iteration, the training dataset is the set of all possible couples of solutions in S_i
- The labelling is inferred by the preferences of the user
- Once O is built, one might use different strategies to pick a new solution
- **Strategy 1**: pick a new solution that is predicted (according to O) to be better than most of the solutions that are already proposed. That is, one that maximizes $\sum_{s' \in S_i} O(s, s')$.

The Proposed Approach

- At each iteration, the training dataset is the set of all possible couples of solutions in S_i
- The labelling is inferred by the preferences of the user
- Once O is built, one might use different strategies to pick a new solution
- **Strategy** 1: pick a new solution that is predicted (according to O) to be better than most of the solutions that are already proposed. That is, one that maximizes $\sum_{s' \in S_i} O(s, s')$.
- **Strategy** 2: pick a new solution that is predicted to be better than most of the solutions that are not proposed. That is, one that maximizes $\sum_{s' \in S \setminus S_i} O(s, s')$.
- . . .

- If O is a valid order then:

$$\forall s_1, s_2, O(s_1, s_2) = -O(s_2, s_1)$$

Tough Challenge

- If O is a valid order then:

$$\forall s_1, s_2, O(s_1, s_2) = -O(s_2, s_1)$$

- There is a simple trick: train O only on couples that are ordered lexicographically. Then use the following prediction rule to answer the question "Is s_1 better than s_2 ?"
  - If $s_1 <_{lex} s_2$ then return $O(s_1, s_2)$
  - Otherwise, return $-O(s_2, s_1)$

Experimental Study

Stable Matching: Decision Version

- Two sets of agents (men, women)
- Each woman ranks the men in a strict order of preferences
- Each man ranks the women in a strict order of preferences
- The purpose to find a complete matching M such that there exists no pair of agents that prefer each other to their partners in M

Stable Matching: Decision Version

- Two sets of agents (men, women)
- Each woman ranks the men in a strict order of preferences
- Each man ranks the women in a strict order of preferences
- The purpose to find a complete matching M such that there exists no pair of agents that prefer each other to their partners in M

Objective Function

- Let M be a stable matching
- Let $Weight_w$ be the sum of the ranks of each woman's partner in M
- Let $Weight_m$ be the sum of the ranks of each man's partner in M
- **Balanced stable matching: minimize** $max(Weight_w, Weight_m)$

Experimental Setup

- Instances size: $\{50, 60, 70\}$

Experimental Setup

- Instances size: $\{50, 60, 70\}$
- Five random instances per size. Each of which has more than 60 solutions

Experimental Setup

- Instances size: $\{50, 60, 70\}$
- Five random instances per size. Each of which has more than 60 solutions
- For each instance, five simulations are conducted

- Instances size: $\{50, 60, 70\}$
- Five random instances per size. Each of which has more than 60 solutions
- For each instance, five simulations are conducted
- Each simulation uses a different set of random solutions of size 40

Experimental Setup

- Instances size: $\{50, 60, 70\}$
- Five random instances per size. Each of which has more than 60 solutions
- For each instance, five simulations are conducted
- Each simulation uses a different set of random solutions of size 40
- The number of iterations is bounded to 10

Experimental Setup

- Instances size: $\{50, 60, 70\}$
- Five random instances per size. Each of which has more than 60 solutions
- For each instance, five simulations are conducted
- Each simulation uses a different set of random solutions of size 40
- The number of iterations is bounded to 10
- We consider the worst possible scenario where the first two solutions are the worst among the sampled solutions

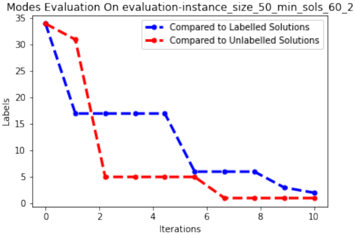
Experimental Setup

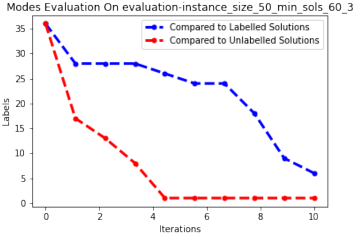
- Instances size: $\{50, 60, 70\}$
- Five random instances per size. Each of which has more than 60 solutions
- For each instance, five simulations are conducted
- Each simulation uses a different set of random solutions of size 40
- The number of iterations is bounded to 10
- We consider the worst possible scenario where the first two solutions are the worst among the sampled solutions
- We use decision trees as prediction models. We make sure that O is a perfect tree at each iteration

Experimental Setup

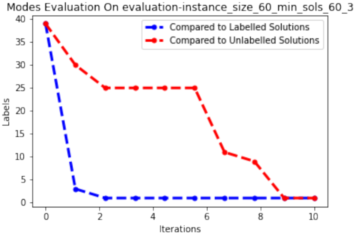
- Instances size: $\{50, 60, 70\}$
- Five random instances per size. Each of which has more than 60 solutions
- For each instance, five simulations are conducted
- Each simulation uses a different set of random solutions of size 40
- The number of iterations is bounded to 10
- We consider the worst possible scenario where the first two solutions are the worst among the sampled solutions
- We use decision trees as prediction models. We make sure that O is a perfect tree at each iteration
- The preferences manager in implemented in Python. It uses the latest version of CP-Optimizer and scikit-learn

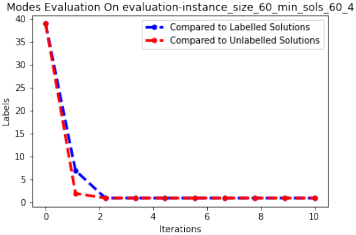
Evaluation of Instances of Size 50



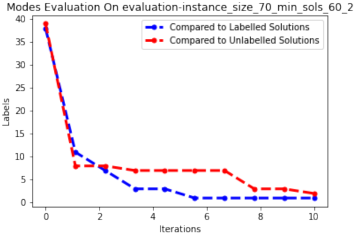


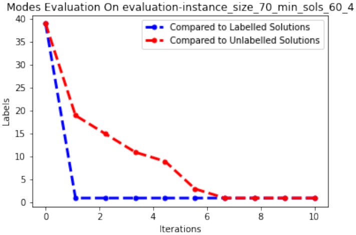
Evaluation of Instances of Size 60





Evaluation of Instances of Size 70





Conclusion

- New framework for interactive CP
- The interactions with the user are limited
- Only ranking queries
- No restriction on the objective function
- Flexible to be used in multiple scenarios
- A lot to explore
- ...

The author would like to thank Bryan Chen, Alice Devilder, Mohamed yassine LOULOU, and Brenda TONLEU NGUISSI for participating in the establishment of a preliminary version of this research.

This work is funded by the “**Centre International de Mathématiques et Informatique de Toulouse (CIMI)**”.
https://www.cimi.univ-toulouse.fr/en/

Thank you!

References I

Benabbou, N. and Lust, T. (2019). An interactive polyhedral approach for multi-objective combinatorial optimization with incomplete preference information. In Amor, N. B., Quost, B., and Theobald, M., editors, *Scalable Uncertainty Management - 13th International Conference, SUM 2019, Compiègne, France, December 16-18, 2019, Proceedings*, volume 11940 of *Lecture Notes in Computer Science*, pages 221–235. Springer.

Bessiere, C., Koriche, F., Lazaar, N., and O'Sullivan, B. (2017). Constraint acquisition. *Artificial Intelligence*, 244:315–342. Combining Constraint Solving with Mining and Learning.

Bessiere, C., Raedt, L. D., Guns, T., Kotthoff, L., Nanni, M., Nijssen, S., O'Sullivan, B., Paparrizou, A., Pedreschi, D., and Simonis, H. (2016). The inductive constraint programming loop. In Bessiere, C., Raedt, L. D., Kotthoff, L., Nijssen, S., O'Sullivan, B., and Pedreschi, D., editors, *Data Mining and Constraint Programming - Foundations of a Cross-Disciplinary Approach*, volume 10101 of *Lecture Notes in Computer Science*, pages 303–309. Springer.

Dechter, R. and Dechter, A. (1988). Belief maintenance in dynamic constraint networks. In Shrobe, H. E., Mitchell, T. M., and Smith, R. G., editors, *Proceedings of the 7th National Conference on Artificial Intelligence, St. Paul, MN, USA, August 21-26, 1988*, pages 37–42. AAAI Press / The MIT Press.

Demirovic, E., Stuckey, P. J., Bailey, J., Chan, J., Leckie, C., Ramamohanarao, K., and Guns, T. (2019). Predict+optimise with ranking objectives: Exhaustively learning linear functions. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 1078–1085. International Joint Conferences on Artificial Intelligence Organization.

Toffano, F., Garraffa, M., Lin, Y., Prestwich, S. D., Simonis, H., and Wilson, N. (2022). A multi-objective supplier selection framework based on user-preferences. *Ann. Oper. Res.*, 308(1):609–640.