# Optimization in Interactive Constraint System Based on Expected Preferences

Hong Huang Barry O'Sullivan

Insight
Centre for Data Analytics


## Abstract

Constraint satisfy problem (CSP) is very useful in formulating many real-life problems. The flexibility of soft constraints in CSP allows the user to express the degree of satisfaction of a constraint, rather than satisfaction or violation in hard constraints.

The key to solving CSP that Involving soft constraints is to acquiring constraint and solution preferences to find a solution that satisfies the problem the most. Many efficient techniques and frameworks have been proposed for finding the best solutions for the constraint satisfaction problem. However, in many real-life problem users cannot express their preferences over subjects accurately and instantly. An example of such situations is when users can only express partial preference or conflict with their other decisions. Also, less attention has been paid to the cost of satisfying such a problem.

In this paper, the problem is based on an interactive framework where the user can express preferences over constraints and solutions, and the sequence of solutions. We consider a simple algorithm to limit the cost of adapting the system to meet user's expression, in addition to search for the best products that satisfy user's demand.


## Introduction

For many real-life problems, there are a lot of recent works have proven that constraints are very useful. In some of the interaction frameworks (e.g. [1]), where combining the standard soft constraint solver and learning modules, it is allowed that users post preferences both over constraints and over solutions. Many techniques and extension of classical CSP have been proposed and developed to make CSP more useful and flexible in describing the real-life problems, which is making CSP more powerful and reliable in solving the problem of acquiring the users' preference in such an interactive system. Some well-studied examples of the extension of classical CSP are Weighted Partial MAX-SAT (WPMS) and Fuzzy CSP (FCSP), etc.

## Algorithm

It is a simple version of our algorithm, but one can give a brief idea of how the interactive constraint system work and react to the feedback given by the user.

**Algorithm 1:** optimize the preference to meet user's requirement

**Data:** R; $\backslash\backslash R$ is the requirement from the user.
Integer: $m$, $n$; $n \in \{1, ..., m\}$; Integer: $i$, $j$; $j \in \{1, ..., i\}$;
$\backslash\backslash m$ is the total number of sols; $i$ is the total number of cons.
Constraint: $C_1, ..., C_i$; Number: $P^1_{C_1}, P^1_{C_2}, ..., P^m_{C_i} \in [0, ..., 1]$;
**Result:** Number: $P'^1_{C_1}, P'^1_{C_2}, ..., P'^m_{C_i} \in [0, ..., 1]$;
Boolean: $F^1_{C_1}, F^1_{C_2}, ..., F^m_{C_i} \in \{0, 1\}$;
**Objective function:** *minimize:* $F^1_{C_1} + F^1_{C_2} + ... + F^m_{C_i}$; *S.T*
$\mathrm{getOrder}(\mathrm{getPreference}(P'^1_{C_1}, ..., P'^1_{C_i}), ..., \mathrm{getPreference}(P'^m_{C_1}, ..., P'^m_{C_i})) == \mathrm{getOrder}(R)$;
$(1 - F^1_{C_1}) * P^1_{C_1} == (1 - F^1_{C_1}) * P'^1_{C_1}$;
$\vdots$
$(1 - F^1_{C_i}) * P^1_{C_i} == (1 - F^1_{C_i}) * P'^1_{C_i}$;
$\vdots$
$(1 - F^m_{C_i}) * P^m_{C_i} == (1 - F^m_{C_i}) * P'^m_{C_i}$;
**if** $\mathrm{getVariable}(C^n_j) \equiv \mathrm{getVariable}(C^{n'}_j)$ **then**
  $P^n_j \equiv P^{n'}_j$; $F^n_j \equiv F^{n'}_j$; $P'^n_j \equiv P'^{n'}_j$;

## Experiment

In each iteration, the system gives top $k$ ($k = 10$ corresponding to the following experiment result) solutions to the user and asks for user's feedback. We recorded the runtime during the acquiring process. After the acquiring process, we synchronized the preference value of constraints for the rest of data. Then, all the solutions are re-rank in order to find the best $k$ solutions which are going to be used for asking the user's satisfaction. The acquiring process will stop when it finds all the optimal solution to the user or after $n$ ($n = 10$ in our experiment) iterations.

Table: Benchmarks and results obtained.

| Benchmark Details | | | Results | | | | | |
|---|---|---|---|---|---|---|---|---|
| #Total | #Opt | Sparse | Only Sols Preference | | | Both Sols & Single Art | | |
| Items | Sols | Factor | #Iters | Time(s) | $Eva$ | #Iters | Time(s) | $Eva$ |
| 1000 | 1 | 10 | 10 | 1.776 | 0.0432 | **2.8** | 1.829 | **0.098** |
| 1000 | 1 | 20 | 10 | 1.807 | 0.0184 | **8.4** | 1.793 | **0.0268** |
| 1000 | 5 | 10 | **8.6** | 1.827 | **0.0686** | **6.4** | 1.843 | **0.0924** |
| 1000 | 5 | 20 | **7.4** | 2.979 | **0.0318** | 1 | 1.631 | **0.0626** |
| 1000 | 15 | 10 | **8.8** | 1.947 | **0.2426** | **4.6** | 1.859 | **0.3700** |
| 1000 | 15 | 20 | **4** | 2.067 | **0.2514** | 1 | 1.843 | **0.3059** |
| 5000 | 1 | 10 | 10 | 8.914 | 0.0329 | 10 | 9.254 | 0.0264 |
| 5000 | 1 | 20 | 10 | 8.102 | 0.0264 | **6.4** | 7.898 | **0.0086** |
| 5000 | 5 | 10 | 10 | 9.052 | 0.0985 | **8.2** | 8.920 | **0.1393** |
| 5000 | 5 | 20 | 10 | 8.433 | 0.0580 | **4** | 7.133 | **0.0730** |
| 5000 | 15 | 10 | 10 | 8.846 | 0.2480 | **8.2** | 8.782 | **0.3087** |
| 5000 | 15 | 20 | 10 | 7.633 | 0.2736 | **8.2** | 6.622 | **0.3431** |
| 10000 | 1 | 10 | 10 | 22.402 | 0.0096 | 10 | 24.281 | 0.0097 |
| 10000 | 1 | 20 | 10 | 18.073 | 0.0149 | 10 | 17.128 | 0.0163 |
| 10000 | 5 | 10 | 10 | 22.581 | 0.0404 | 10 | 22.472 | 0.1615 |
| 10000 | 5 | 20 | 10 | 19.220 | 0.0645 | **8.2** | 16.990 | **0.1689** |
| 10000 | 15 | 10 | **8.2** | 22.023 | **0.2672** | **8.2** | 22.522 | **0.3573** |
| 10000 | 15 | 20 | 10 | 16.343 | 0.2347 | **8.2** | 16.917 | **0.2706** |

The datasets in our experiments are all randomly generated. The items in these datasets are all containing 7 attributes and 4 constraints over the attributes. The deviation index of results are represented as $Eva$ in Table 1. The results showed in Table 1 are the mathematical average of the results of 5 experiment runs. The calculation for the deviation index in our experiments is: $Eva = \frac{1}{n}\sum_{i=1}^{n}\frac{(O_i - i)}{m}$ (where $n$ is the number of the Oracle optimal solutions, $m$ is the number of total items, $O_i$ is the index of solution $i$ in the optimal order with respect to all the items). For the results, lower the deviation index value means better solutions it found.

## Future work

- Study different scenarios in order to tackle the problem better and reduce the runtime and iterations.
- Theoretically estimate the least iterations that are required to find the optimal solutions or restore the optimal order of all solutions respected to user's preferences.

## References


Rossi, F., Sperduti, A.: Acquiring both constraint and solution preferences in interactive constraint systems. Constraints 9(4), 311–332 (2004)


A World Leading SFI Research Centre

This work has been supported by a research grant by Science Foundation Ireland under grant number SFI/12/RC/2289