Sven Löffler, Ke Liu and Petra Hofstedt ({sven.loeffler, Liuke, hofstedt}@b-tu.de)

Optimizing Constraint Satisfaction Problems by Regularization for the Sample Case of the Warehouse Location Problem

Brandenburg University of Technology Cottbus - Senftenberg

Motivation

The Complexity of Constraint Problems

Since the search space of constraint satisfaction problems (CSPs), and consequently also the solution time, is very big, we are always interested in a speed-up of the solution process. There are various ways to describe a CSP in practice and consequently, the problem can be modeled by different combinations of constraints, which results in differences in resolution speed and behavior.

Hence, the diversity of models and constraints for a given CSP offers us an opportunity to improve the problem solving process by using another model in which a subset of constraints is replaced with a constraint which combines the original ones but offers a faster solution process.

The Improvement Idea

The solution speed and behavior of a CSP depends amongst other things on the number of backtracks in the depth-first search of the solution process and redundancy in the propagation of constraints.

We developed a new approach for the optimization of CSPs, where singleton constraints or sets of constraints are substituted by regular constraints which are combined and minimized. The aim of this CSP reformulation is on the one hand to reduce slowing-down redundancy in constraints over shared variables and on the other hand to remove origins of backtracks in the search.

Definitions

Constraints are predicate logical formulas, e.g.

$X \in \{1, ..., 9\}$, $Y = 3.7$, $Z \leq Y$

for describing and efficient solving of problems with incomplete information.

"Holy Grail of programming" (E.C. Freuder)

The **Sum constraint** $sum(X, R, r)$ takes an array of variables $X = \{x_1, ..., x_n\}$, a relation $R \in \{<, \leq, =, >, \geq, \neq\}$ and a result variable r as input. Its successful propagation ensures that the sum of the variables in X is in relation R to the result r.

$$sum(X, R, r) = (\sum x_i)\ R\ r$$

The **IfThen constraint** $ifThen(c_{if}, c_{then})$ takes two constraints c_{if} and c_{then} as input and guarantees that, if the first constraint c_{if} is fulfilled, also the second constraint c_{then} is fulfilled.

$$ifThen(c_{if}, c_{then}) = satisfied(c_{if}) \rightarrow c_{then}$$

The **Scalar constraint** $scalar(X, C, R, r)$ takes an array of variables $X = \{x_1, ..., x_n\}$, an array of corresponding integer coefficients $C = \{c_1, ..., c_n\}$, a relation $R \in \{<, \leq, =, >, \geq, \neq\}$ and a result variable r as input. Its successful propagation ensures that the scalar product of X and C is in relation R to the result r.

$$scalar(X, C, R, r) = (\sum_{i=1}^{n} x_i * c_i)\ R\ r$$

CSP Constraint Satisfaction Problem $P = (X, D, C)$

$X = \{x_1, x_2, \ldots, x_n\}$	(set of variables)
$D = \{D_1, D_2, \ldots, D_n\}$	(finite domains, D_i is the domain of x_i)
$C = \{c_1, c_2, \ldots, c_m\}$	(Each constraint c_i is a relation defined over a subset of the variables X)

A solution of a CSP P is a complete instantiation satisfying all constraints of the CSP P.

A constraint satisfaction optimization problem (**CSOP**) $P^{opt} = (X,D,C,f)$ is defined as a CSP with an optimization function f that maps each solution to a numerical value to be minimized or maximized, respectively.

A **Directed Acyclic Graph** (DAG) is a DFA $M = (Q, \sum, \delta, q_{initial}, F)$, where the states $q \in Q$ are partitioned into levels $Q = \{Q_1, ..., Q_n\}$, $Q_i = \{q_{i,1}, q_{i,2}, ...\} \mid i \in \{1, ..., n\}$, the initial state $q_{initial}$ is the only element of Q_1, the final states are the members of the last level of states ($F = Q_n$), and transitions are only allowed from a state $q_{i,j}$ in level Q_i to a state $q_{i+1,k}$ in level Q_{i+1}.

The **Regular Membership Constraint** obtains a DAG $M = (Q, \sum, \Delta, q_0, F)$ and an ordered set of variables $\{x_1, x_2, \ldots, x_n\} = X$ with domains $D = \{D_1, D_2, \ldots, D_n\}$, $\forall\ i \in \{1, \ldots, n\}$, $D_i \in \sum$ as input and guarantees that every sequence $d_1d_2...d_n$ of values for $x_1, x_2, \ldots, x_n$ must be a word of the regular language recognized by the DAG M.

$$regular(X, M) = \{ w_1, w_2, \ldots, w_n \mid \forall i \in \{1,\ldots,n\}, w_i \in D_i, (w_1, w_2, \ldots, w_n) \in L(M)\}.$$

Regularization

Scalar and Sum

A given scalar constraint $scalar(X, C, R, r)$ can be replaced by an regular constraint $regular(X, M)$ with DAG $M = (Q, \sum, \delta, q_{initial}, F)$.

- $Q = \{Q_0, ..., Q_{n+1}\}$, where Q_0 contains only $q_{0,0}$, Q_{n+1} contains only $q_{n+1,0}$
  - In each state $q_{i,j} \in Q \setminus \{q_{n+1,0}\}$ the j represents the partial scalar product $\sum x_k * c_k$, where each x_k has the value of the k-th edge on the path $q_{0,0} \rightarrow q_{i,j}$.
- $\sum$ = the union of the domains D of the variables X
- δ = is a function $Q \times \sum \rightarrow Q$
  - Let $i \in \{0, ..., n\}$ then it holds $\delta(q_{i,p}, s) = q_{i+1,p+s*c}$, if s is a possible value for the scalar product $scalar(X, C, R, r)$.
  - It holds $(q_{n,p}, s) = q_{n+1,0}$ if p is in relation R with s
- $q_{initial} = q_{0,0}$
- $F = \{q_{n+1,0}\}$

The sum constraint $sum(X, R, r)$ can be interpreted as special version of the scalar constraint, where the coefficients are all equal to 1 ($c_i = 1$, $i \in \{1, ..., n\}$). Thus, we can use the same transformation as for the scalar constraint $scalar(X, \{1, ..., 1\}, R, r)$.

If then

We consider only $ifThen(c_{if}, c_{then})$ constraints, where

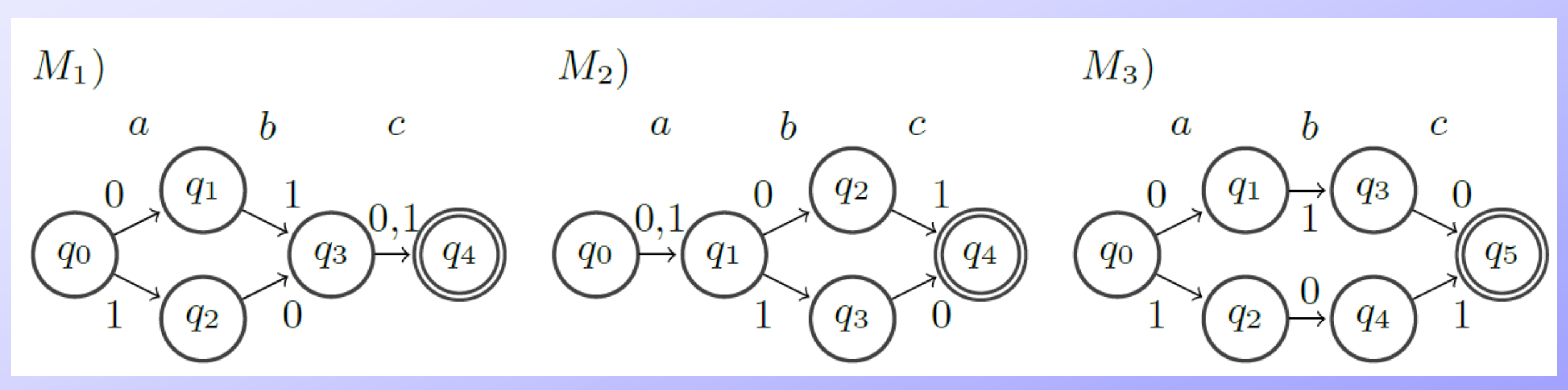
- the variables of both involved constraints are disjoint: $scope(c_{if}) \cap scope(c_{then}) = \{\}$.
- transformations for c_{if} and c_{then} into regular constraints (c_{rif} and c_{rthen}) exist.

Let be M_{if} the DAG used in c_{rif} and M_{then} the DAG used in c_{rthen}.

$$M_{ifThen} = (M_{if} \circ M_{then}) \cup (M_{\neg if} \circ M_{all})$$

A regular constraint, which uses the automaton M_{ifThen}, where $M_{\neg if}$ is the complement automaton of M_{if} and M_{all} is an automaton which accepts all words which have the same length as the accepted words of M_{then}, can be used as a replacement for the $ifThen(c_{if}, c_{then})$ constraint.

The Benefit of Regularization - Removing (Unwanted) Redundancy

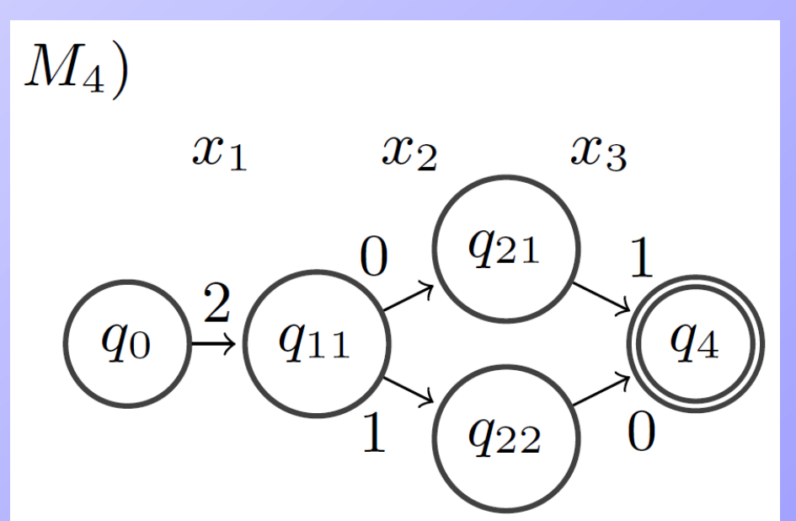


The reduction of states and transitions. M_1 and M_2 have together 10 states and 12 transitions. M_3, the intersected automaton of M_1 and M_2, has only 6 states and 6 transitions.

The Benefit of Regularization - Removing Backtracks

Consider a CSP $P = (X,D,C)$, where $X = \{x_1, x_2, x_3\}$, $D = \{D_1, D_2, D_3 \mid D_1 = \{0, 1, 2\}, D_2 = D_3 = \{0, 1\}\}$ and $C = \{c_1, c_2, c_3\}$, with $c_1 = (x_1 \neq x_2)$, $c_2 = (x_1 \neq x_3)$ and $c_3 = (x_2 \neq x_3)$.

Some search strategies set x_1 to value 0 or 1, which cannot satisfy the CSP, so backtracking is necessary. If the well-known *alldifferent* constraint is used, then no backtracking is necessary. But what happens if we change c_1 to $x_1 > x_2$? It is no longer obvious that we can substitute c_1, c_2 and c_3 with an *alldifferent* constraint. But we still have the situation that common search strategies assign 1 to x_1, so that backtracking is necessary again.



The substitution of the constraints c_1, c_2 and c_3 by one regular constraint $c_r = \{\{x_1, x_2, x_3\}, M_4\}$ removes backtracking from the search. The DAG M_4 is created by the intersection of the three automatons of the regular representations of c_1, c_2 and c_3.

The Warehouse Location Problem

The Model

$P = (X, D, C, f)$, where $X = X^{ws} \cup X^{wc} \cup X^{sc} \cup \{x_{tc}\}$, $D = D^{ws} \cup D^{wc} \cup D^{sc} \cup \{D_{tc}\}$, $C = C^{scalar} \cup C^{counts} \cup C^{countw} \cup C^{ifThen} \cup \{c^{sum}\}$ and $f = minimize(x_{tc})$.

The variables in X^{ws} are binary variables represent, whether a warehouse i supplies a store j (set to 1) or not (set to 0).

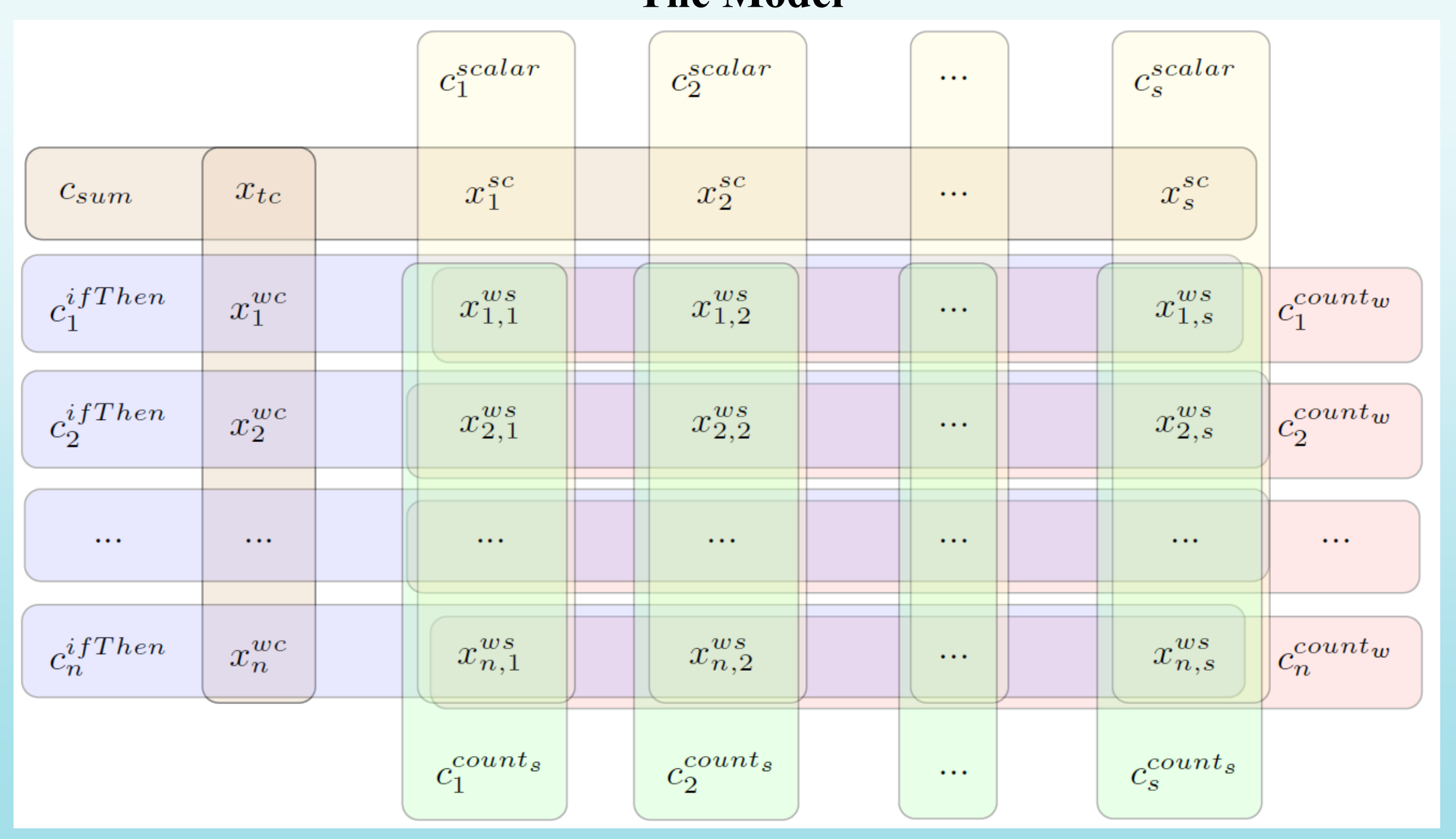
Each binary variable in X^{wc} with domain $D^{wc} = \{0, f_c\}$ describes the costs for warehouse i, whether it is used (set to f_c) or not (set to 0).

Each variable in X^{sc} describes the cost of a store based on the information by which warehouse it is supplied. For example, if store i is supplied by warehouse j then the i-th variable in X^{sc} is instantiated by $M_{i,j}$, where the matrix M represents the supply cost of each store to supply a warehouse.

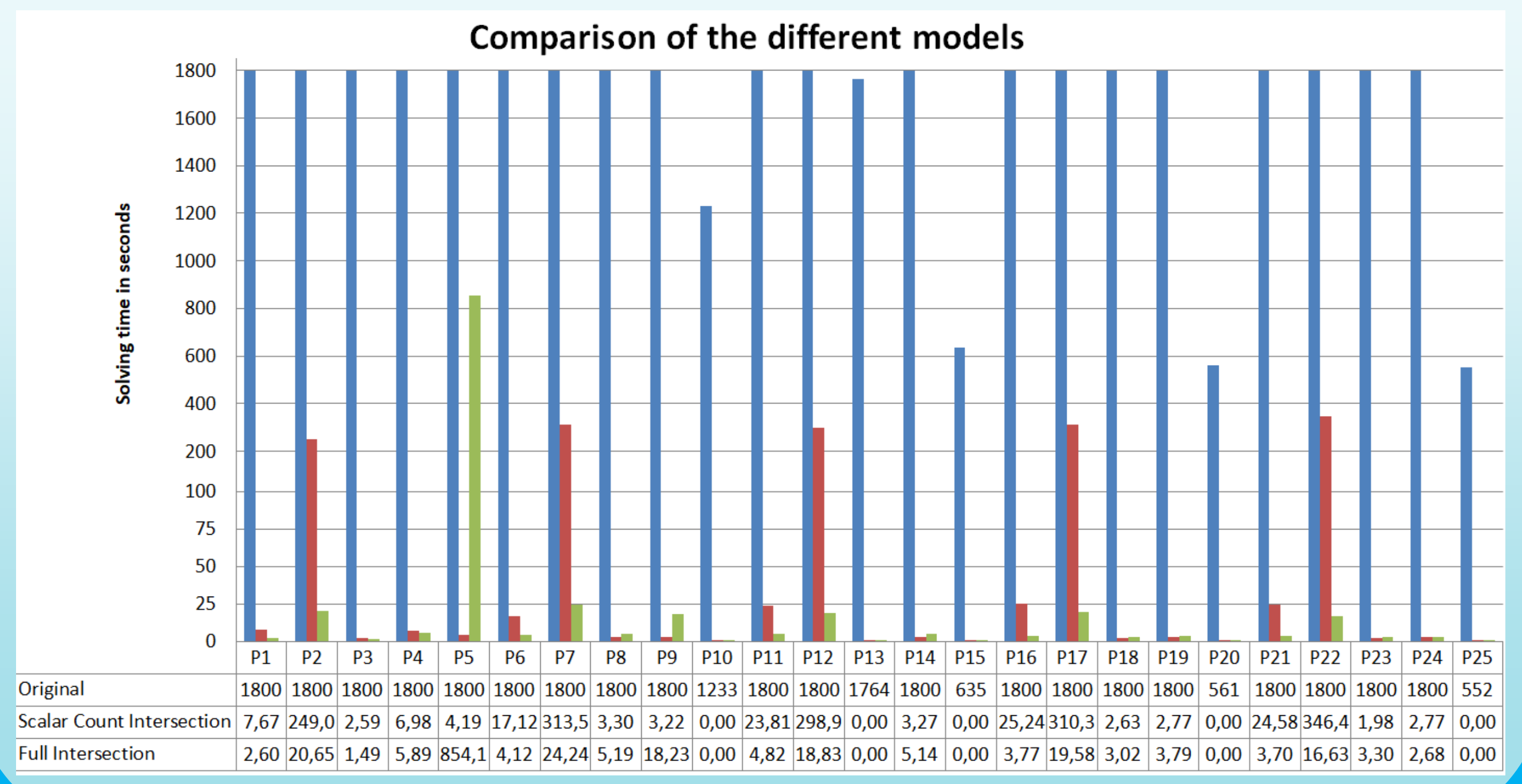
The variable x_{tc} with domain $D_{tc} = \{0, ..., infinity\}$ describes the total costs for suppling all stores. The goal is it to minimize these costs ($f = minimize(x_{tc})$).

$$\begin{array}{llll}
C^{scalar} & = \{c_j^{scalar} & = scalar(x_{*,j}^{ws}, M_{*,j}, =, x_j^{sc}) & \mid \forall j \in \{1, ..., s\}\} \\
C^{count_s} & = \{c_j^{count_s} & = count(1, \{x_{*,j}^{ws}\}, =, 1) & \mid \forall j \in \{1, ..., s\}\} \\
C^{count_w} & = \{c_i^{count_w} & = count(0, \{x_{i,*}^{ws}\}, \geq, (s - c_i)) & \mid \forall i \in \{1, ..., n\}\} \\
C^{ifThen} & = \{c_i^{ifThen} & = ifThen(sum(x_{i,*}^{ws}, \neq, 0), (x_i^{wc} = f_c)) & \mid \forall i \in \{1, ..., n\}\} \\
c_{sum} & & = sum((X^{sc} \cup X^{wc}), =, x_{tc}) &
\end{array}$$

The Model



The Results



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25
Original	1800	1800	1800	1800	1800	1800	1800	1800	1800	1233	1800	1800	1764	1800	635	1800	1800	1800	1800	561	1800	1800	1800	1800	552
Scalar Count Intersection	7,67	249,0	2,59	6,98	4,19	17,12	313,5	3,30	3,22	0,00	23,81	298,9	0,00	3,27	0,00	25,24	310,3	2,63	2,77	0,00	24,58	346,4	1,98	2,77	0,00
Full Intersection	2,60	20,65	1,49	5,89	854,1	4,12	24,24	5,19	18,23	0,00	4,82	18,83	0,00	5,14	0,00	3,77	19,58	3,02	3,79	0,00	3,70	16,63	3,30	2,68	0,00

Future Work

Future work includes a comparison and potentially an integration with work, e.g. the tabulation transformation of CSPs from [1]. Furthermore, more direct transformations from global constraints to the regular constraint must be developed. The approach of direct transformations must be merged with the approach of general regularization for small sub CSPs.

[1] Akgün, Ö., Gent, I.P., Jefferson, C., Miguel, I., Nightingale, P., Salamon, A.Z.: Automatic discovery and exploitation of promising subproblems for tabulation. In: Hooker, J.N. (ed.) Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, Proceedings. LNCS, vol. 11008, pp. 3-12. Springer (2018). https://doi.org/10.1007/978-3-319-98334-9 1