Generalizing Global Constraints Based on Network Flows

Igor Razgon[1,2], Barry O'Sullivan[1,2], and Gregory Provan[2]

[1] Cork Constraint Computation Centre, University College Cork, Ireland
[2] Department of Computer Science, University College Cork, Ireland
{i.razgon|b.osullivan|g.provan}@cs.ucc.ie

Abstract. Global constraints are used in constraint programming to help users specify patterns that occur frequently in the real world. In addition, global constraints facilitate the use of efficient constraint propagation algorithms for problem solving. Many of the most common global constraints used in constraint programming use filtering algorithms based on network flow theory. We show how we can formulate global constraints such as GCC, Among, and their combinations, in terms of a tractable set-intersection problem called Two Families Of Sets (TFOS). We demonstrate that the TFOS problem allows us to represent tasks that are often difficult to model in terms of a classical constraint satisfaction paradigm. In the final part of the paper we specify some tractable and intractable extensions of the TFOS problem. The contribution of this paper is the characterisation of a general framework that helps us to study the tractability of global constraints that rely on filtering algorithms based on network flow theory.

1 Introduction

Global constraints are used in constraint programming to help users specify patterns that occur frequently in the real world (see, for example, [8–13]). In addition, global constraints facilitate the use of efficient constraint propagation algorithms for problem-solving. Many of the most common global constraints used in constraint programming employ filtering algorithms based on network flow theory. Examples are the Global Cardinality Constraint (GCC) [11], and the Among Constraint [3], which generalize a number of other global constraints such as NotAllEqual, Max, and Member constraints.

A real-life problem usually needs *combinations* of global constraints, rather than a single constraint. Some of these combinations may be efficiently solved, but generally they are intractable [13]. A question that arises in this context is to describe a tractable problem that can represent various combinations of network flow-based global constraints. This question is addressed in this paper. In particular, our contributions are as follows.

1. We define a model that includes a ground set V and two families F_1 and F_2 of subsets of V. Any two elements of each family are either disjoint or contained one in the other. Each subset of V contained in these families is associated with two nonnegative integers called minimal and maximal cardinalities. We refer to the model as TFOS, which is an acronym for Two Families Of Sets.

2. Given a TFOS model (V, F_1, F_2) we say that a subset of V is valid if the size of its intersection with each set contained in F_1 or F_2 lies between the cardinalities assigned to that set. We define a TFOS problem as finding the largest valid subset of V and study the tractability of the problem.
3. We show that GCC and Among constraints, as well as (some of) their combinations considered in [13], can be represented by the TFOS model. In the proposed representation, V is the set of all values of the CSP being considered, F_1 is the family of all domains, F_2 represents the global constraints. By introducing additional sets to F_1, we demonstrate that it is possible to represent as a TFOS problem some optimization tasks that seem difficult to express as a classical CSP.
4. We propose a propagation algorithm that can be used to speed up search in cases where part of some intractable problem is presented as a TFOS problem. The propagation algorithm is based on the approach suggested in [11].
5. We discuss possible extensions of the TFOS model. In particular we show tractability of the weighted TFOS problem. Then we prove that introducing an additional family of sets of V with the same properties as the families of the TFOS model makes the resulting problem NP-hard. Finally, we show that we can preserve polynomial solvability by restricting the properties of the third family.

Several other approaches to the design of generalized global constraints are described in [2, 4, 13]. The work reported in [13] is most closely related to our approach: it proves the tractability of two types of combinations of GCC and Among constraints. However, the TFOS model is more general because, as we show further in this paper, it can express the combinations of constraints considered in [13] as well as a number of additional combinations of constraints that seem hard to express by that approach. The authors of [2] propose a method to design tractable logical combinations of some primitive constraints. However, their method is unable to express some basic network flow-based global constraints such as Among. The approach described in [4] has an emphasis different from ours: it aims at the design of a unifying language for expressing global constraints, but this language does not necessarily enforce tractability.

The rest of the paper is organized as follows. Section 2 provides the necessary background. Section 3 defines the TFOS problem and proves its tractability. Section 4 describes possible applications of the proposed model. Section 5 provides a scheme for developing a propagation algorithm for TFOS. Section 6 discusses possible extensions of the model. The implications of the TFOS model on the modelling process are discussed in Section 7. A number of concluding remarks are made in Section 8.

2 Background

Given a directed graph $G = (V, E)$ with two specified nodes s and t called *source* and *sink*, a flow in G is a function from the set of arcs $E(G)$ to the set of non-negative integers[1] that satisfies the following conditions: for each vertex v except s and t, the amount of flow entering v equals the amount of the flow leaving v, the amount of the flow entering s as well as the amount of flow leaving t is 0.

[1] Generally, a flow does not have to be integral but this restriction is sufficient here.

The maximum flow problem, in its simplest formulation, associates non-negative integer capacities with each edge of G and asks for the maximum flow from s to t such that the flow delivered through each edge does not exceed its capacity. The problem can be solved by picking an initial flow and augmenting it iteratively by finding a path from s to t in the *residual* graph obtained from G by removing some edges and adding "opposites" to other edges (see [1], Sections 6.3 and 6.4 for a detailed description). The time required by each iteration is proportional to the sum of the number of vertices and edges of G. The flow, due to its integrality, is augmented by at least one at each iteration. Hence, the resulting complexity is the complexity of one iteration multiplied by the maximum flow. Polynomial-time algorithms for the maximum flow are well-known [1].

The constraint satisfaction problem (CSP) is defined on a set of variables $VAR = \{var_1, \ldots, var_n\}$ and a set values $VAL = \{val_1, \ldots, val_m\}$. Each variable has a *domain*, which is a subset of VAL. The objective is to assign each variable with exactly one value from its domain subject to certain *constraints*. A constraint specifies a subset S of variables and restricts tuples of values allowed to be assigned to the variables of S. The set S is called the *scope* of the constraint. The scope of a *global* constraint may be of an arbitrary size, even including all the variables. In this paper we consider two types of global constraints: the Global Cardinality Constraint (GCC) [11] and the Among constraint [3]. The former constraint specifies for each value of VAL its minimal and maximal number of occurrences in a solution of the given CSP. The latter constraint specifies for a subset T of VAL the minimal and maximal number of occurrences of values of T in a solution of the given CSP.

The CSP is intractable in general but there have been many tractable classes studied (see, for example, [6]). In particular, a CSP constrained by a single GCC or a single Among constraint is tractable because it can be transformed into a network flow problem [13].

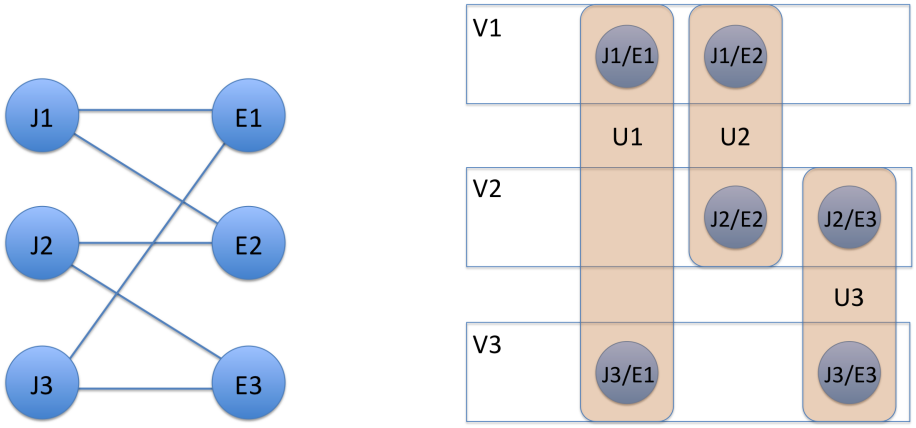
3 The TFOS Model

Let V be a set of vertices. Let F_1 and F_2 be two families of nonempty subsets of V such that any two sets that belong to the same collection are either disjoint or contained one in the other. The intersection between sets from different families may be arbitrary. Each set $Y \in F_1 \cup F_2$ is associated with two non-negative numbers called *minimal and maximal cardinalities* that do not exceed $|Y|$. We refer to the model (V, F_1, F_2) as TFOS, which is an abbreviation of Two Families Of Sets. Let X be a subset of V such that for each element Y of F_1 or F_2, the size of $X \cap Y$ lies between the cardinalities associated with Y[2]. We call X a *valid* subset of V. The task of the *TFOS problem* is to find the largest valid subset of V. Consider the following example of application of the proposed model.

Example 1. Consider a scheduling problem with sets J and E of jobs and employees, respectively. Each job is specified by a subset of employees who can perform this job.

[2] If $Y \in F_1 \cap F_2$ and, consequently, Y is associated with two distinct pairs of cardinalities, one for F_1, the other for F_2, this condition should be satisfied for both pairs of cardinalities.

Each employee is specified with the minimum and the maximum number of jobs to be performed. The task is to assign each job with exactly one employee so that no employee violates her (or his) restriction of the minimal and maximal allowed number of jobs.



(a) An example problem. (b) The corresponding TFOS model.

Fig. 1. An illustration of Example 1.

In Figure 1 we have three jobs and three employes. Figure 1(a) shows which job can be performed by which employee: the vertex denoting a job is adjacent to the vertices denoting the employees that can perform this job. The corresponding families of sets are shown in Figure 1(b). In particular, one family of sets denotes sets V_1, V_2, V_3, each of them includes Job/Employee pairs with the same first element. The second family of sets includes sets U_1, U_2, and U_3 which unite the pairs according to the same second element.

More formally, let (V, F_1, F_2) be a TFOS model such that V is the set of all pairs (J_i, E_k) where E_k is an employee who can perform job J_i. Assume that there are n jobs and m employees. Then F_1 contains n subsets of V and the i-th subset contains all pairs with the first element J_i. The cardinalities for each set of F_1 are both 1, expressing the requirement that exactly one employee is to be assigned to a job. The family F_2 contains m subsets of V with the k-th subset containing all pairs having E_k as the second element. The cardinalities of the subset corresponding to employee E_k are the minimal and the maximal number of jobs allowed to E_k. It is not hard to observe that any feasible solution of the specified TFOS problem represents a valid assignment of jobs to employees. Observe that the resulting TFOS model is equivalent to a CSP with a GCC constraint, where F_1 represent domains of variables and F_2 represent cardinality constraints assigned to values. ▲

Now we prove the tractability of the TFOS problem. Given a TFOS model (V, F_1, F_2), we assume that both F_1 and F_2 cover all vertices of V. If, for example, the set $V \setminus \bigcup F_1$

is not empty, we can add it to F_1 accompanied with cardinalities 0 and $|V \setminus \bigcup F_1|$. Clearly, the resulting TFOS problem is equivalent to the original one.

Let $F_1 = \{S_1, \ldots, S_m\}$, $F_2 = \{T_1, \ldots, T_k\}$. We define the graph $G(F_1, F_2)$ as follows. The vertices of the graph are $s, t, s_1, \ldots, s_m, t_1, \ldots, t_k$, where s_i and t_i correspond to the respective sets, s and t are the source and the sink of the flow. There is an edge (s, s_i), for every S_i which is maximal in F_1 and an edge (t_i, t) for every T_i which is maximal in F_2. There is an edge (s_i, s_j) whenever S_j is a maximal subset of S_i and an edge (t_j, t_i) whenever T_j is a maximal subset of T_i. Finally, let $V(S_i, T_j) \subseteq S_i \cap T_j$ be the set of all u such that S_i and T_j are minimal in their families subject to containing u. There is an edge (s_i, t_j) whenever $V(S_i, T_j)$ is not empty.

Observation 1 *We make the following observations.*

1. *$G(F_1, F_2)$ has exactly one edge entering any s_i and exactly one edge leaving any t_i.*
2. *$G(F_1, F_2)$ has $O(|V|)$ vertices and $O(|V|)$ edges.*

Proof. See Appendix A.

Now we associate with each edge of $G(F_1, F_2)$ its minimal and maximal capacities. The edge entering any s_i or leaving any t_j is associated with the respective minimal and maximal cardinalities of the corresponding set. Finally, the minimal capacity of any edge between s_i and t_j is 0, the maximal capacity is $|V(S_i, T_j)|$.

We will prove that the size of the largest valid subset of V equals the amount of the maximal flow that can be delivered from s to t in $G(F_1, F_2)$. The proof is divided into two lemmas (see Appendix A). In the first one we show that for any valid $X \subseteq V$, there is a flow of size $|X|$. The other lemma shows that for any flow from s to t there is a valid set X whose size equals the amount of the delivered flow. Combining these two lemmas together yields the desired result.

Theorem 1. *Given a TFOS model (V, F_1, F_2), the problem of finding the largest valid subset of V can be solved in $O(|V|^2)$. (In some cases there may be no valid subset at all. In this case, a network flow algorithm reports the absence of feasible flow.)*

Proof. The maximum flow in graph $G(F_1, F_2)$ is at most $|V|$, and all the capacities are integral. Consequently, a traditional iterative approach to solving the maximum flow problem with maximal and minimal capacities (see [1], Section 6.7) solves the problem in $O(|V|)$ iterations. The complexity of each iteration is proportional to the sum of the number of vertices and the number of edges of $G(F_1, F_2)$, which is $O(|V|)$ from Observation 1. ■

An illustration of an application of Theorem 1 is presented in Figure 2. In this figure we have a binary CSP: ellipses represent variables, black circles represent the values, and the edges represent the binary conflicts. In this particular example all the conflicts form cliques. So, we can introduce two families of sets $\{V_1, V_2, V_3, V_4\}$ where for each family at least one value has to be selected and the family $\{U_1, \ldots, U_5\}$ of cliques where at most one value can be selected. According to Theorem 1, this CSP can be solved in polynomial time.

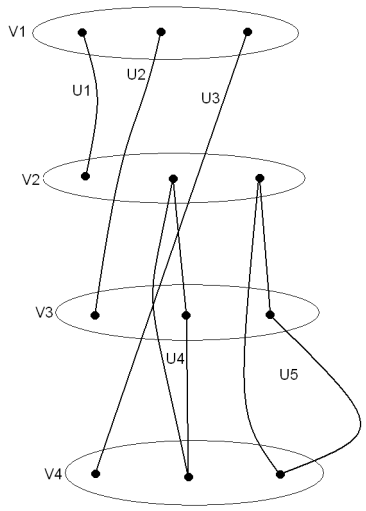


Fig. 2. An illustration of an application of Theorem 1.

Remark. If a TFOS model represents a CSP with n variables and the maximum domain size d then V corresponds to the set of all values of the CSP and has size $O(nd)$; the maximum flow corresponds to a solution of the CSP which has size n. Hence the flow algorithm takes $O(n^2d)$.

4 Applications

This section shows that the GCC and the Among constraints, as well as (some of) their combinations, can be represented by the TFOS model. Moreover, it shows how to represent some optimization tasks as a TFOS problem that are hard to express using the classical CSP paradigm.

4.1 GCC and Value-Disjoint Among Constraints

Consider the scheduling problem described in Example 1. We specify additional requirements for this problem. Assume that the employees are partitioned according to their professions, based on the minimal and the maximal number of jobs allowed to be performed by persons of each profession. It is not hard to update the TFOS model shown in Example 1 so that it expresses the new requirement. For each profession P, add to F_2 a set that includes all pairs (J_i, E_k) such that E_k has profession P. Set the minimal and maximal cardinalities of that set equal to the minimal and maximal number of jobs, respectively, allowed to fellows of profession P.

It can be shown that this TFOS model is equivalent to a CSP with a combination of GCC and value-disjoint Among constraints [13]. In particular, the value-disjoint Among constraints are represented by the new sets added to F_2. The domains of variables and the cardinality constraints are represented as shown in Example 1.

The resulting TFOS model can be further updated to express new requirements. For example, imagine that the jobs specified in our example scheduling problem span some period of time, say, a week, i.e. the existing constraints restrict the number of jobs in a week. In addition we can restrict the number of jobs performed by each particular person in a day. Let $P_1, \ldots, P_7$ be the partition of jobs according to the day they are to be performed. For each employee E_k, we add to F_2 seven new subsets, the j-th subset contains all elements (J_i, E_k) where J_i is a job of P_j that can be performed by E_k. The cardinalities of j-th subset are the minimal and the maximal number of jobs allowed for E_k on j-th day. Observe that the new elements of F_2 preserve the property of the TFOS model.

4.2 GCC and Variable-Disjoint Among Constraints

In the scheduling problem presented in Example 1, assume that only a subset $F \subseteq E$ of employees is constrained by restricting the minimal and maximal number of jobs. Assume, as in the previous subsection, that the jobs of J span a time period of a week and we constrain the number of jobs performed by *all* the employees in a day. That is, we add seven new restrictions to the example problem that specify the minimal and the maximal number of jobs performed by the employees of F in each of the seven days of a week. These new restrictions can be represented as variable-disjoint Among constraints. It can be shown that the resulting problem is equivalent to the combination of GCC and Among constraints if the number of partition classes of J is not necessarily seven, but an arbitrary integer.

The description of the obtained scheduling problem in terms of the TFOS model is not straightforward, because the sets corresponding to the variable-disjoint Among constraints cannot be added to any of the families of sets defined in Example 1 without violating the properties of these families. To obtain the description, observe that there are three possible cases of the example scheduling problem. In the first case, for each job there is an "unconstrained" employee that can perform this job. This case is trivial because each job can be assigned to such an employee without violating any constraint. In the second case, $E = F$, that is, there are no unconstrained employees at all. In this case, partition the sets in F_1 into seven classes according to the day the corresponding jobs are assigned. The Among constraints can be expressed by unions of the sets that belong to the same partition class. The cardinalities for each set are the minimal and the maximal number of jobs allowed on the corresponding day. The sets corresponding to Among constraints can be added to F_1 because they do not violate the required property that any two sets of F_1 are either disjoint or contained one in the other.

In the last case of the example scheduling problem, unconstrained employees can perform only a part of the required jobs but not all. In this case, the TFOS model is constructed in two stages. In the first stage, we take the TFOS model described in Example 1 and replace F_1 by the "projection" of F_1 to F. That is, we replace each set in F_1 by a subset that contains all the elements (J_i, E_k) such that $E_k \in F$. We associate the sets that have lost some of their elements as a result of this replacement with cardinalities 0 and 1, the cardinalities of the other sets both remain 1. In the second stage the Among constraints are introduced by analogy with the previous case. It can be shown that if the resulting TFOS problem does not have a feasible solution, the example scheduling

problem does not have a solution either. Otherwise, take any valid set of the obtained TFOS model and assign the jobs "unassigned" by this set to respective unconstrained employees. The resulting assignment is a solution to the problem.

4.3 Compact CSP Modeling

Consider again the scheduling problem and the corresponding TFOS model described in Section 4.1. Recall that the sets of family F_1 represent domains of the corresponding CSP. A relative inflexibility of CSP as a modelling language is that each variable must be assigned with exactly one value from its domain. In this subsection we demonstrate that the TFOS model does not have this disadvantage. In particular we show that by introducing a number of additional subsets into family F_1 of the TFOS model being considered, we can represent a scheduling problem that seems difficult to be expressed in terms of classical CSP.

Recall that the scheduling problem described in Section 4.1 assigns exactly one employee to each job and that the set of all employees is partitioned according to their profession. Assume that we would like to assign each job not with a single employee but with a crew of employees with a specified minimal and maximal number of fellows of each profession participating in the crew. To introduce these new constraints into the TFOS model, we partition each set S of F_1 according to the profession of the employees with which the elements of S correspond. We associate each partition class with the cardinalities equal to the minimal and the maximal number of fellows of the corresponding profession allowed to be assigned to the job corresponding to S. Then we add the resulting new sets to F_1.

5 Towards a Propagation Algorithm for TFOS

In this section we present a Propagation Algorithm (PA) for the TFOS problem. The input of the algorithm consists of a TFOS model (V, F_1, F_2) and an integer k. If there is no valid subset of V of size at least k, the algorithm reports infeasibility. Otherwise, it outputs the subset of V containing the elements that *do not* belong to any valid subset of size at least k. Before presenting the PA itself, let us specify how it can be used to speed up the search.

We assume that an optimization problem is formulated as finding a largest subset of V subject to certain restrictions, a part of which are families F_1 and F_2 with their minimal and maximal cardinalities, and that the problem is solved by a systematic Search Algorithm (SA). In every iteration, the SA possesses additional data: the size m of the largest known subset of V satisfying all the restrictions and a subset V' of V. The SA tries to extend V' to a size of at least $m + 1$. The PA decides whether the extension is possible if the only restrictions considered are those imposed by families F_1 and F_2. If yes, the PA specifies which elements of $V \setminus V'$ cannot participate in such a set. Clearly, if the PA reports infeasibility, the SA must backtrack immediately. If a set of infeasible elements is specified, the SA discards them in its attempt to extend V', thus pruning the branches of the search tree. Note that the PA considers the TFOS model $(V \setminus V', F_1', F_2')$, where the elements of F_1' and F_2' are obtained from F_1 and

F_2, respectively, by appropriate restriction of their sets and updating cardinalities. In particular, a set $S \in F_1$ is transformed into a set $S \setminus V'$ of F_1' with the cardinalities $max(l(S) - |S \cap V'|, 0)$ and $u(S) - |S \cap V'|$, where $l(S)$ and $u(S)$ are the cardinalities of S in F_1. The transformation from F_2 to F_2' is analogous. The minimal size of a valid subset "pursued" by the PA is $m - |V'| + 1$. (If the problem being considered is a CSP, the minimal size of a valid subset always equals to the number of unassigned variables.)

Given a TFOS model (V, F_1, F_2) and an integer k as input, the PA proceeds in two stages. In the first stage, the feasibility of a valid subset of size k is checked. In particular, the PA constructs a graph $G'(F_1, F_2)$ obtained from $G(F_1, F_2)$ by adding a new node s' and an additional edge (s', s) of capacity k. Then the maximum flow from Y s' to t is computed. Clearly a valid subset of size k is feasible only if a flow of size k can be delivered from s' to t. If a valid set of the required size is found out to be feasible, the algorithm goes on to the second stage: computing the subset of infeasible values of V.

Proposition 1. *Let u be a value of V. Let $F_1(u)$ and $F_2(u)$ be the minimal elements of F_1 and F_2 that contain u. Let e be an edge of $G'(F_1, F_2)$ from the node corresponding to $F_1(u)$ to the node corresponding to $F_2(u)$.[3] There is a feasible subset X of V such that $|X| \geq k$ and $u \in X$ if and only if there is flow of size at least k from s' to t and the flow delivered through edge e is nonzero.*

Thus, the set of the infeasible elements of V can be extracted in $O(|V|)$ from the set of *infeasible* edges of $G'(F_1, F_2)$, i.e., the edges that are left "untouched" by any maximum flow from s' to t. These edges can be found by an approach suggested in [11]. According to that approach we consider the residual graph G_R obtained from $G'(F_1, F_2)$ by delivering flow Y. The graph G_R is partitioned into strongly connected components. The infeasible edges are those whose ends do not belong to the same component. Partitioning into strongly connected components for G_R can be done in $O(|V|)$ applying an algorithm by Tarjan (see, for example, [5]). Hence the complexity of the propagation algorithm is determined by the time complexity of the maximum flow computation, which is $O(|V|^2)$ by Theorem 1. Finally, note that the if the TFOS model being considered represents a CSP, the complexity of the PA, which, in terms of CSP, is called *achieving generalized arc-consistency*, is $O(n^2d)$.

6 Extensions of the TFOS model

6.1 The Weighted TFOS Problem

Given a TFOS model (V, F_1, F_2) and a weight function w associating each element of V with a weight, the task of the weighted TFOS problem is to find the largest valid subset of V having the smallest weight (the weight of a set is computed as the sum of weights of its elements). By analogy with the unweighted TFOS problem, it can be shown that the weighted TFOS problem generalizes various network flow-based global constraints with costs [12].

[3] There is such an edge because $u \in V(F_1(u), F_2(u))$, hence $V(F_1(u), F_2(u))$ is not empty and thus corresponds to an edge by construction of $G(F_1, F_2)$.

Theorem 2. *The weighted TFOS problem is tractable.*

Proof. The weighted TFOS problem can be transformed into the problem of finding the minimum cost flow ([1], Chapter 10) in a graph $G_w(F_1, F_2)$ that can be obtained from $G(F_1, F_2)$ by introducing the following modifications. The edges entering the nodes corresponding the elements of F_1 or leaving the nodes corresponding to the elements of F_2 are associated with zero costs. Each edge between a node corresponding to a set $A \in F_1$ and a node corresponding to a set $B \in F_2$ is split into $|V(A, B)|$ edges corresponding to elements of $V(A, B)$. The edge corresponding to each element $u \in V(A, B)$ is associated with cardinalities 0 and 1 and with cost $w(u)$. The correctness of the transformation can be proved in a way similar to that employed in Lemmas 1 and 2 (presented in the appendix). ■

6.2 Three Families of Sets cause NP-hardness

If we allow more than two families with the property of families of a TFOS model, the corresponding optimization problem can be shown to be NP-hard.

Theorem 3. *An extension of the TFOS problem that includes three families of sets is NP-hard.*

Proof. The NP-hardness can be shown by the reduction from a version of 3-SAT where each variable occurs at most one as a positive literal and and most twice as a negative literal. This problem is well known to be NP-complete (see, for example, the classical "Computational Complexity" book of Papadimitriou). Let F be a 3-CNF formula over a set of variables $v_1, \ldots v_n$ such that each v_i appears at most once and each $\neg v_i$ appears at most twice. Let Z be a binary CSP with variables corresponding to the clauses of F, the values of the domain of each variable correspond to the literals of the respective clause, two values are incompatible if they correspond to the positive and the negative literal of the same variable. It is not hard to see that Z is soluble if and only if F is satisfiable.

Observe that Z has two types of conflicts. A conflict of the first type can be called an *isolated* conflict. It involves a pair of values (val_1, val_2) which are incompatible but no other value is incompatible with val_1 nor with val_2. A conflict of the second type can be called a *complex* conflict. It involves a triple (val_1, val_2, val_3) such that val_1 is incompatible with both val_2 and val_3 and no other value is incompatible with either of val_1, val_2, val_3.

We introduce the three families of sets structure (U, F_1, F_2, F_3) as follows. The set U includes all the domain values of Z (the values of different domains are considered distinct). F_1 is the family of all domains. The set F_2 is constructed as follows. For each isolated conflict (val_1, val_2), the set $\{val_1, val_2\} \in F_2$. For each complex conflict (val_1, val_2, val_3), the set $\{val_1, val_2)\}$ belongs to F_2. No other sets are contained in F_3. The set F_3 contains only sets $\{val_1, val_3\}$ for each complex conflict (val_1, val_2, val_3). It is not hard to observe that the sets within each family are pairwise disjoint. The associate the upper and lower bounds 1 with the sets of F_1, upper bound 1 and lower bound 0 are associated with the sets of F_2 and F_3.

Observe that (U, F_1, F_2, F_3) has a valid set of size $n = |F_1|$ if and only Z is soluble. Indeed, if Z is soluble then any solution of Z is a valid set of (U, F_1, F_2, F_3) because it has exactly one value within each domain (i.e. each set of F_1) and contains no pairs of conflicting values (i.e has at most one value within each set of F_2 and F_3). Conversely, any valid set has exactly one value in each domain and satisfies all the constraints of Z, hence it is a solution of Z. ■

6.3 Preserving Tractability: Restrictions on the third family

Although the optimization problem based on three families of sets is NP-hard in general, it could be solved efficiently if we restrict the properties of the third family. The following example demonstrates this possibility. We define the three family of sets problem (*3FOS*) as a four tuple (V, F_1, F_2, F_3), where the first three components are the same as in the TFOS model and F_3 is a family $\{Y_1, \ldots Y_l\}$ of subsets of i such that $Y_i \subset Y_j$ whenever $i > j$. Assume that the minimal cardinalities associated with the elements of F_3 are all zeros. Let $u_1, \ldots u_l$ be the respective maximal cardinalities. We may assume that $u_i < u_j$ whenever $i > j$ as if not, the cardinality constraint imposed by u_i is redundant.

Theorem 4. *The* 3FOS *problem can be solved efficiently using an algorithm for the weighted TFOS problem as a procedure.*

Proof. We associate the elements with weights as follows. All elements of $V \setminus Y_1$ are associated with zero costs. All elements of $Y_1 \setminus Y_2$ are associated with some large positive weight W, say 1000. For $i > 1$, let K be the weight associated with the elements of $Y_{i-1} \setminus Y_i$. Then the elements of $Y_i \setminus Y_{i+1}$ (or Y_i in case $i = l$) are associated with weight $K * u_{i-1}/u_i$. Observe that $X \subseteq V$ violates the cardinality constraints imposed by F_3 if and only if the weight of X is greater then $W * u_1$. This observation suggests the following way of solving the problem.

Solve the weighted TFOS problem (V, F_1, F_2) with the weights assigned as shown below. If there is no feasible valid subset of V then the original problem has no feasible valid subset either. If the resulting largest valid subset X has a weight smaller than or equal to $W * u_1$ then X is a solution of the original problem. Otherwise, we learn that the original problem has no solution of size $|X|$. To introduce this additional constraint, we add to F_1 set V with cardinalities 0 and $|X|-1$. (If such a set already exists, we adjust its maximum cardinality.) Then we solve the resultant weighted TFOS model again. This process may be repeated a number of iterations. It stops if in some iteration a feasible solution of weight at most $W * u_1$ is found or infeasibility is reported. Otherwise, the maximal allowed cardinality of set V in family F_1 is decreased by 1. Infeasibility is reported if this maximal cardinality has been reduced to 0 with no feasible solution found before. ■

Although the described example is rather artificial, it it shows the existence of a non-trivial polynomially solvable optimization problem based on more than two families of sets.

7 Modelling Intractable Problems using the TFOS Model

In this section we demonstrate that the TFOS model can be useful for modelling intractable problems. The main benefit of the TFOS model is that it allows us to model intractable problems in a way that makes constraint propagation more efficient. An intractable problem is usually modelled as a *conjunction* of global constraints [13, 14]. Each constraint in the conjunction is propagated separately in a number of iterations until no value can be removed from the domains of the constrained variables. Hence, the number of constraints in the conjunction has a multiplicative factor on the complexity of the propagation algorithm and it is desirable that the number of such constraints be as small as possible. We show now that there are hard problems that can be modelled using a much smaller number of TFOS structures as compared to other types of global constraints.

Consider, for example, the problem obtained at the end of Section 4.1. One can imagine that this problem represents one *shift* of some timetabling problem that can occur in real applications. A timetabling problem usually consists of a number of shifts with intersecting sets of jobs associated with different shifts that make the problem hard [7]. It follows from the description in Section 4.1 that such a hard timetabling problem can be modelled using *one* TFOS structure per shift, while the number of GCC and Among constraints is linear in the number of variables participating in a shift (a linear number of among constraints is needed, for instance, to represent constraints added at the end of Section 4.1). Moreover, the separate propagation of the GCC and Among constraints cannot discard all the values that might be discarded by the propagation of TFOS structures.

8 Conclusion

We have presented an optimization problem that we termed the TFOS problem. We showed that various combinations of network flow-based global constraints can be expressed in terms of the TFOS problem. We also demonstrated that the TFOS model can describe scenarios that seem difficult to express in terms of a classical CSP. We presented global constraints in terms of a scheduling problem rather than an abstract setting, which demonstrated that the TFOS model can be useful for modelling sophisticated resource allocation tasks. We identified some tractable and intractable extensions of the TFOS model. In particular, we showed that the weighted TFOS problem is tractable and that with three families of sets, though intractable in general, it can be made tractable by applying restrictions on the properties of the third family. Finally, we discussed the implications of the TFOS model on the modelling process.

While this paper has raised a route for generalizing collections of flow-based global constraints, much remains to be done. Firstly, it would be interesting to implement our proposed framework and compare its efficiency against more standard approaches to solving collections of flow-based global constraints. Secondly, while we have outlined a propagation scheme for TFOS, a generic filtering algorithm must be developed in order to make the approach more general. A possible direction of further theoretical research is to identify other tractable generalizations of the TFOS problem and to design methods

for coping with intractability (such as approximation or parameterized algorithms) for intractable extensions of the TFOS problem.

Acknowledgements

Razgon and O'Sullivan are supported by Science Foundation Ireland (Grant No. 05/IN/I886). Provan is supported by Science Foundation Ireland (Grant No. 04/IN3/I524).

References

1. R. Ahuja, T. Magnatti, and J. Orlin. *Network Flows*. Prentice Hall, 1993.
2. F. Bacchus and T. Walsh. Propagating logical combinations of constraints. In *IJCAI*, pages 35–40, 2005.
3. N. Beldiceanu and E. Contjean. Introducing global constraints in CHIP. *Mathematical and Computer Modelling*, 12:97–123, 1994.
4. C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. The range and roots constraints: Specifying counting and occurrence problems. In *IJCAI*, pages 60–65, 2005.
5. T. Cormen, C. Leiserson, R. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.
6. R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.
7. A. Meisels and A. Schaerf. Modelling and solving employee timetabling problems. *Annals of Mathematics and Artificial Intelligence*, 39:41–59, 2003.
8. G. Pesant. A regular language membership constraint for finite sequences of variables. In *CP*, pages 482–495, 2004.
9. C.-G. Quimper, A. Lopez-Ortiz, P. vanBeek, and A. Golynski. Improved algorithms for the global cardinality constraint. In *Principles and Practice of Constraint Programming-CP2004*, pages 542–556, Toronto, Canada, sep 2004. Springer.
10. J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proceedings of AAAI*, pages 362–367, 1994.
11. J.-C. Régin. Generalized arc consistency for global cardinality constraint. In *AAAI/IAAI, Vol. 1*, pages 209–215, 1996.
12. J.-C. Régin. Arc consistency for global cardinality constraints with costs. In *CP99*, pages 390–404, 1999.
13. J.-C. Régin. Combination of among and cardinality constraints. In *CPAIOR 2005*, pages 288–303, 2005.
14. J.-C. Régin and Jean-Francois Puget. A filtering algorithm for global sequencing constraints. In *CP97*, pages 32–46, 1997.

A Intermediate Proofs from Observation 1 to Theorem 1

Observation 1 *We make the following observations.*

1. $G(F_1, F_2)$ *has exactly one edge entering any* s_i *and exactly one edge leaving any* t_i.
2. $G(F_1, F_2)$ *has* $O(|V|)$ *vertices and* $O(|V|)$ *edges.*

Proof. Each observation is proven separately.

1. If S_i is maximal then (s, s_i) is the only edge that enters s_i. If S_i is not maximal, assume that there are two sets S_j and S_f such that S_i is a maximal subset of both of them. From the structure of F_1, either $S_f \subseteq S_j$ or $S_j \subseteq S_f$. The former case contradicts S_i being the maximal set contained in S_j. In the latter case, there is the same contradiction regarding S_i and S_f. The proof for t_i is symmetric.
2. The statement regarding the number of vertices easily follows from the observation that the number of sets in F_1 as well as in F_2 is at most $2 * |V| - 1$. This observation can be proven by induction on $|V|$. It is immediate if $|V| = 1$. For $|V| > 1$, F_1 may contain (in the worst case) V itself and a partition of V into n, ($n \geq 2$) subsets of sizes $y_1, \ldots y_n$. By the induction hypothesis, i-th partition class together with all its subset sum up to at most $2 * y_i - 1$. Summing these numbers together we get $2 * |V| - n$ subsets that together with V itself are at most $2 * |V| - 1$ subsets.
   To prove the upper bound on the number of edges, observe that there is at most one edge entering a node corresponding to an element of F_1 and at most one edge leaving a node corresponding to an element of F_2. It follows that we need to check only the number of edges connecting the nodes corresponding to elements of different families. To this point note that each of these edges corresponds to a non-empty subset of V and that the subsets associated with different edges do not intersect. ■

Lemma 1. *Let X be a valid subset of V. Then there is a flow of size $|X|$ from s to t.*

Proof. We construct the flow as follows. Associate every edge (s_i, t_j) with the flow $|X \cap V(S_i, T_j)|$. Having associated the edges (s_i, t_j) with the appropriate flows, proceed as follows. Whenever there is vertex s_i such that all the edges leaving s_i have already been associated with their flows and the edge entering s_i has not been yet, associate the edge entering s_i with the flow equal to the sum of flows on the edges leaving s_i. Repeat analogously for t_i with the only difference that the edge leaving t_i is associated with the sum of flows on the edges entering t_i. The obtained assignment of flows guarantees that the flow entering s as well as the flow leaving t is zero and that the flow entering each intermediate node equals the flow leaving that node. In the rest of the proof, we show that the constructed flow "respects" all the capacities and has size $|X|$.

Claim. The flow entering any s_i equals $|S_i \cap X|$ and the flow leaving any t_j equals $|T_j \cap X|$.

Proof. We prove the claim regarding $s_1, \ldots, s_m$; the proof regarding $t_1, \ldots, t_k$ is symmetric. Assume that $s_1, \ldots, s_m$ are ordered in such a way that $i < j$ whenever $S_i \subseteq S_j$. The proof is by induction on this sequence. Let $e_1, \ldots, e_l$ be the edges leaving s_1. Due to minimality of S_1 the heads of the edges correspond to sets $T'_1 \ldots T'_l$ of F_2. Denote $X \cap V(S_1, T'_i)$ by X'_i. Observe that $X'_1, \ldots, X'_l$ is a partition of $X \cap S_1$. Indeed, any two X'_y and X'_z are disjoint because any $u \in X'_y \cap X'_z$ implies that one of T'_y and T'_z is contained in the other one contradicting the minimality assumption for the larger set. For any $u \in X \cap S_1$, the set S_1 is the minimal one that contains u just because it is a minimal set in F_1. Let T' be the minimal set in F_2 that contains u. Clearly, there is an edge between the vertices corresponding to S_1 and T', hence there exists a j such that $T' = T'_j$ and u belongs to X'_j. We have proved that $X'_1 \ldots X'_l$ are disjoint and cover all the vertices of $X \cap S_1$. Hence, they form a partition of $X \cap S_1$. Consequently,

$|S_1 \cap X| = |X'_1| + \ldots + |X'_l|$. Recall that $|X'_j|$ is exactly the flow assigned to e_j and the flow on the edge entering S_1 is the sum of flows on all e_j. The validity of the claim for S_1 follows immediately.

Consider now s_i for $i > 1$. Let $e_1, \ldots, e_l$ be the edges leaving s_i. The head of every e_j is either some s_y or some t_z. In the former case, let $X'_j = S_y \cap X$, in the latter case let $X'_j = V(S_i, T_z) \cap X$. Similar to the case with S_1, we can show that $X'_1, \ldots X'_l$ form a partition of $X \cap S_i$. Observe that the flow assigned to every e_j is exactly $|X'_j|$: if the head of e_j is some s_y, it follows from the induction hypothesis taking into account that $y < i$; if the head of e_j is some t_z, the observation follows from the initial assignment of flows. Taking into account that $|S_i \cap X| = |X'_1| + \ldots + |X'_l|$ and that the flow on the edge entering S_i is the sum of flows on the edges leaving S_i, $|S_i \cap X|$ is exactly the flow on the edge entering s_i. □

Considering that X is a valid set, it respects the minimal and the maximal cardinalities of all the sets in F_1 and F_2. It follows that the flow assigned to the edge entering each s_i is valid and leaving each t_j is valid. For the edges of type (s_i, t_j), the validity follows by definition of the flow on these edges.

It remains to show that the amount of flow delivered from s to t is exactly $|X|$. To this point observe that the amount of flow leaving s is the sum of flows entering the nodes corresponding to the maximal sets of F_1. Let $S'_1, \ldots, S'_l$ be these maximal sets. Clearly, $|S'_i \cap X|$ is exactly the flow entering each vertex s'_i. Taking into account that each element of X belongs to some S'_i, we obtain $X = |S'_1 \cap X| + \ldots |S'_l \cap X|$. ■

Lemma 2. *Let F be a valid flow from s to t. Then there is a valid set X such that $|X|$ equals the amount of flow that leaves S.*

Proof. For every edge e between vertices s_i and t_j, fix $X(e) \subseteq V(S_i, T_j)$ such that $|X(e)|$ is the amount of flow on edge (s_i, t_j). There is such an $X(e)$ since the maximal capacity on the edge is $|V(S_i, T_j)|$. Let X be the union of all $X(e)$.

Using the inductive argument analogous to the one used in proof of Lemma 1, we can show that the flow entering every s_i and leaving every t_j equals $|S_i \cap X|$ and $|T_j \cap X|$, respectively. Taking into account that the flow is valid, the number of elements of every S_i and T_j contained in X satisfies their minimal and maximal cardinalities. ■