Symmetric Cardinality Constraint with Costs

Waldemar Kocjan[1], Per Kreuger[2], Björn Lisper[1]

[1] Mälardalen University, Västerås, Sweden
{waldemar.kocjan,bjorn.lisper}@mdh.se
[2] Swedish Institute Of Computer Science, Kista, Sweden
piak@sics.se

Abstract. The symmetric cardinality constraint is described in terms of a set of variables $X = \{x_1, \ldots, x_k\}$, which take their values as subsets of $V = \{v_1, \ldots, v_n\}$. It constraints the cardinality of the set assigned to each variable to be in an interval $[l_{x_i}, u_{x_i}]$ and at the same time it restricts the number of occurrences of each value $v_j \in V$ in the sets assigned to variables in X to be in an other interval $[l_{v_j}, u_{v_j}]$. In this paper we extend the symmetric cardinality constraint with a function which associate with each value of each variable a cost and constraints the global cost of the constraint to the sum of costs associated with assigned values. We also give an algorithm for computing the consistency of a symmetric cardinality constraint with costs and describe filtering methods for this constraint.

1 Introduction

The symmetric cardinality constraint, introduced in [1], is specified in terms of a set of variables $X = \{x_1, \ldots, x_k\}$, which take their values as subsets of $V = \{v_1, \ldots, v_n\}$. The cardinality of the set assigned to each variable is constrained by an interval $[l_{x_i}, u_{x_i}]$, where l_{x_i} and u_{x_i} are non–negative integers. In addition, it constraints the number of occurrences of each value $v_j \in V$ in the sets assigned to variables in X to be an interval $[l_{v_j}, u_{v_j}]$. Both l_{v_j} and u_{v_j} are non–negative integers.

The symmetric cardinality constraint is an extension of the global cardinality constraint [2] on sets. While global cardinality constraint allows us to model instances of matching problems where one variable can be matched with only one value, but the number of values in matching must be in a given interval, the symmetric cardinality constraint enables to assign to a variable a set of values with a cardinality described by an interval. The global cardinality constraint can be modeled using symmetric cardinality constraint by restraining the cardinality of a set assigned to any variable to 1.

Later the global cardinality constraint was extended with a cost function ([3, 4]), which makes possible to model some assignments problems. Nevertheless, assignment problems, where a (possibly empty) set of values needs to be assigned to a variable in the problem, can not be handled easily.

Consider the following instance of a project management problem, which consists of a task of assigning a number of workers, with possibly multiple competences, to a set of activities, each requiring a number of workers (possibly none) with specified competences. Due to former experience of the members of the project we can approximate an amount of time necessary for respective worker to accomplish given activity. Moreover, a total amount of time which can be spent on the project is given.

The goal is to produce an assignment, which satisfies the following constraints:

- every member of a project must be assigned to a minimum and a maximum number of activities in the project
- every activity must be performed by a minimum and a maximum number of persons
- each person can be assigned to an activity he/she is qualified to perform and, by symmetry, each activity must be performed by a qualified personnel.
- the total time spend on accomplishing the project must not be exceeded.

Clearly, due to the fact that global cardinality constraint require exactly one value to be assigned to a variable, it can not be used to solve such a problem without introducing auxiliary constraints. On the other hand, we can easily model this problem by extending the symmetric cardinality constraint with a cost function.

In this paper we show how the symmetric cardinality constraint can be extended with a cost function and investigate if the filtering algorithms for the global cardinality constraint with costs can be applied to the symmetric cardinality constraint with cost.

The rest of the paper is organized as follow. The next section, 2, gives some preliminaries on graphs, flows and set constraint satisfaction problem. Section 3 briefly describes the symmetric cardinality constraint. Then, in Section 4 we give a formal definition of the symmetric cardinality constraint with costs and describe the method for achieving its consistency. In the following section, 5, we describe an algorithm for computing a minimum cost flow in a network of the constraint. The following Section 6 describes the filtering method for this constraint. Finally, Section 7 conclude this work.

2 Preliminaries

2.1 Graph

Definitions in this section follows the presentation in [5].

A *directed graph* $G = (N, E)$ consists of a set of nodes (vertices) N and arcs (edges) E, where every arc $(i, j) \in E$ is an ordered pair of distinct nodes. An *oriented graph* is a directed graph having no symmetric pair of arcs.

A graph $G = (N, E)$ is a *bipartite graph* if we can partition its node set into two subsets N_1 and N_2 so that for each arc $(i, j) \in E$ either $i \in N_1$ and $j \in N_2$ or $i \in N_2$ and $j \in N_1$.

A *directed network* is a directed graph whose nodes and/or arcs have associated numerical values. In this paper we do not make distinction between terms "network" and "directed network".

An arc (i, j) connects node i with node j, i.e. in directed graph it is an arc oriented from node i to node j. A *path* in a graph G from v_1 to v_k is a sequence of nodes $[v_1, \ldots, v_k]$ such that each (v_i, v_{i+1}) is an arc for $i \in [1, \ldots, k-1]$. The path is *simple* if all its nodes are distinct. A path is a *cycle* if $k > 1$ and $v_1 = v_k$.

2.2 Flows

Let $G = \{N, E\}$ be a directed network in which each arc $(i, j) \in E$ is associated with two non–negative integers l_{ij} and u_{ij} representing lower and upper bound on a capacity of a flow through (i, j). A flow $f(i, j)$ on arc (i, j) represents the amount of commodity that the arc accommodates. More formally:

Definition 1. *A flow in a network G is a function that assigns to each arc (i, j) of the network a value $f(i, j)$ in such way that*

1. *$l_{ij} \leq f(i, j) \leq u_{ij}$, where l_{ij} and u_{ij} are lower and upper bounds on a capacity of (i, j)*
2. *for each node i in the network G it is true that $\sum_k f(k, i) = \sum_j f(i, j)$*

The second property is known as a conservation law and states that the amount of flow of some commodity incoming to each node in G is equal the amount of that commodity leaving each node.

In this paper we refer to *the feasible flow problem* which resolves if there exists a flow in G which satisfies capacity constraint for all arcs in G.

Moreover, we refer here to *the minimum cost flow problem* in which each arc $(i, j) \in E$ has an associated *cost* denoted as c_{ij}. For any flow f in G

$$cost(f) = \sum_{(i,j) \in E} c_{ij} \times f(i, j) \tag{1}$$

It is a well known fact that if the capacity bounds of a flow problem are integral and there exists a feasible flow for the network, then the maximum and minimum flows between any two nodes flows are also integral on all arcs in the network. Hence, if there exists a feasible flow in a network there also exists an integral feasible flow. Moreover, if there exists an integral feasible flow in G and all costs are integral then by Equation (1) the cost of f is also integral. In this paper when we refer to a feasible flow or a minimum cost flow we always mean an *integral* feasible flow and an *integral* minimum cost flow.

2.3 Set Constraint Satisfaction Problem

A set constraint satisfaction problem [1] is defined as a triple (X, D, Cs) where

- $X = \{x_1, \ldots, x_n\}$ is a finite set of variables.

– $D = \{D_1, \ldots, D_n\}$ is a set of finite sets of elements such that for each i, x_i takes as value a subset of D_i.
– Cs is a set of constraints on the values particular subsets of the variables in X can simultaneously take. Each constraint $C \in Cs$ constrains the values of a subset $X(C) = \{x_{C_1}, \ldots, x_{C_k}\}$ of the variables in X and may be thought of as a subset $T(C)$ of the Cartesian product $= C_{C_1} \times \ldots, \times C_{C_k}$ where each $C_{C_i} = \{C \mid C \subseteq D_{C_i}\}$.

The set constraint satisfaction problem differs from the constraint satisfaction problem as described in [6] by the fact that the value, which can be assigned to any $x_i \in X$, is in the powerset of D_i.

Definition 2. *A set constraint satisfaction problem is consistent if and only if there exists an assignment P with the following properties:*

1. *For each variable $x_i \in X$ with domain D_i, the value $P(x_i)$ assigned to x_i by P must be a subset of D_i.*
2. *For each constraint $C \in Cs$ and each variable in $X(C) = \{x_{C_1}, \ldots, x_{C_k}\}$ the tuple $\langle P(x_{C_1}), \ldots, P(x_{C_k})\rangle \in T(C)$.*

Let $X(C)$ be the set of constraint variables and $D(X(C))$ their domains. A value $v \in D(x_i)$ for x_i is consistent with C iff $\exists P(P(X(C)) \in T(C))$ such that v is an element in the value $P(x_i)$.

A value graph [7] of a constraint C is a bipartite graph $GV(C) = (X(C) \cup D(X(C)), E)$, where $(x, v) \in E$ if and only if $v \in D_x$.

3 Symmetric Cardinality Constraint

The symmetric cardinality constraint is specified in terms of a set of variables $X = \{x_1, \ldots, x_k\}$, which take their values as subsets of $V = \{v_1, \ldots, v_n\}$. The cardinality of the set assigned to each variable is constrained by the interval $[l_{x_i}, u_{x_i}]$, where l_{x_i} and u_{x_i} are non–negative integers. In addition, it constraints the number of occurrences of each value $v_j \in V$ in the sets assigned to variables in X to be an interval $[l_{v_j}, u_{v_j}]$. Both l_{v_j} and u_{v_j} are non–negative integers.

More formally, the symmetric cardinality constraint is defined as follows.

Definition 3. *A symmetric cardinality constraint is a constraint C over a set of variables $X(C)$ which associates with each variable $x_i \in X(C)$ two non-negative integers l_{x_i} and u_{x_i}, and with each value $v_j \in D(X(C))$ two other non-negative integers l_{v_j} and u_{v_j} such that a restriction of an assignment P to the variables in $X(C)$ is an element in $T(C)$ iff*
$\forall i\,(l_{x_i} \leq \#(x_i, P) \leq u_{x_i})$ and $\forall j\,\left(l_{v_j} \leq \#(v_j, C, P) \leq u_{v_j}\right)$.

Consistency of a symmetric cardinality constraint C is achieved by computing a feasible flow in a particular value network $N(C)$ obtained from a value graph of C by

– orienting each edge of $N(C)$ from values to variables. Since each value can occur in a subset assigned to a variable at least 0 and at most 1 time for each arc $(v, x) : l(v, x) = 0, u(v, x) = 1$
– adding a source node s and connecting it which each value. For every arc $(s, v_i) : l(s, v_i) = l_{v_i}, u(s, v_i) = u_{v_i}$
– adding a sink node t and an arc from each variable to t. For each such arc $(x_i, t) : l(x_i, t) = l_{x_i}, u(x_i, t) = u_{x_i}$
– adding an arc (t, s) with $l(t, s) = 0$ and $u(t, s) = \infty$

The complexity of computing the consistency of a symmetric cardinality constraint is the same as the complexity of computing a feasible flow in $N(C)$, which in worst case is the same as for computing a maximum flow in $N(C)$. Computing a maximum flow in a network depends on used algorithm (see [9] for comparison of different maximum flow algorithms), but it can be approximated to $O(n^3)$, where n is a number of nodes in a network. This gives a complexity relative to the number of constraint variables, $|X|$, and the size of their domains $|D(X(C))|$, of $O((|X| + |D(X(C))|)^3)$ time.

Given a flow f in $N(C)$ we can obtain *the residual graph* for f, i.e. the network representing utilization and remaining capacity in the network with respect to f, as follows.

Let $N(C)$ be a value network of C, the residual graph of $N(C)$ with respect to a flow f, denoted by $R(f)$, is a graph with the same set of nodes as $N(C)$. The arc set of $R(f)$ is defined as follows.

– if $f(i, j) < u_{ij}$ then $(i, j) \in R(f)$, $r_{ij} = u_{ij} - f(i, j)$
– if $f(i, j) > l_{ij}$ then $(j, i) \in R(f)$, $r_{ji} = f(i, j) - l_{i,j}$

r_{ij} denotes the residual capacity of (i, j) The residual capacity of a path p, denoted by $r(p)$, is a minimum value r_{ij} for all $(i, j) \in p$.

A value v_j of a variable x_i is not consistent with C if and only if there is no feasible flow in $N(C)$ which contains a flow on arc (v_j, x_i). Given a feasible flow f, a flow on (v_j, x_i) is the same in any feasible flow f' if and only if neither (v_j, x_i) nor (x_i, v_j) is contained in a strongly connected component in $R(f)$ involving at least three nodes (see Theorem 1 in [1]). Thus, if a flow on (v_j, x_i) is equal to 0, and (v_j, x_i) is not contained in a strongly connected component in $R(f)$ involving at least three nodes, then, the value v_j of x_i is not consistent with C.

Generally, there exist a number of algorithms for computing strongly connected components in a graph (see [10] p. 560 for references). The algorithm given in [11] computes strongly connected components in $O(n + m)$ time, where n is a number of nodes and m is a number of edges in a graph. In terms of symmetric cardinality constraint $n = |X(C)| + |X(D(C))|$ and $m = |X(C)| + 2 * |X(D(C))| + 2$, thus, the complexity of filtering the domains of the variables in C is $O(|X| + |X(D(C))|)$ time.

4 Consistency of Symmetric Cardinality Constraint with Costs

The symmetric cardinality constraint with costs extends the symmetric cardinality constraint with a cost function.

Definition 4. *A cost function on a variable set X is a function which associates with each element $v_j \in D(x)$ a non–negative integer denoted by $cost(x_i, v_j)$.*

The following gives a definition of symmetric cardinality constraint with cost as a conjunction of a symmetric cardinality constraint and a sum constraint.

Definition 5. *A symmetric cardinality constraint with cost is a constraint C over a set of variables $X(C)$ which associates with each variable $x_i \in X(C)$ two non-negative integers l_{x_i} and u_{x_i}, with each value $v_j \in D(X(C))$ two other non-negative integers l_{v_j} and u_{v_j} a cost function on each $x_i \in X(C)$ and an integer H such that a restriction of an assignment P to the variables in $X(C)$ is an element in $T(C)$ iff*

- $\forall i \, (l_{x_i} \leq \#(x_i, P) \leq u_{x_i})$
- $\forall j \, \big(l_{v_j} \leq \#(v_j, C, P) \leq u_{v_j}\big)$.
- $\sum_{i=1}^{|X(C)|} cost(x_i, P) \leq H$

where $\#(x_i, P)$ is the cardinality of the set assigned to x_i by P, $\#(v_j, C, P)$ is the number of variables to which v_j is assigned by P and $cost(x_i, P) = \sum_{v_j \in D(x_i)} cost(x_i, v_j)$ for each v_j assigned to x_i by P.

To achieve consistency of a symmetric cardinality constraint with costs we extend the value network of symmetric cardinality constraint, $N(C)$, with the cost function by

- associating with each arc $(v_j, x_i) \in N(C)$ a cost $c_{v_j, x_i} = cost(x_i, v_j)$
- associating with every arc (s, v_j) $c_{sv_j} = 0$
- associating with every arc (x_i, t) $c_{x_i t} = 0$
- associating with (t, s) $c_{ts} = 0$

Note that the value network of a symmetric cardinality constraint with costs is independent of H.

The following proposition defines consistency of a symmetric cardinality constraint with costs.

Proposition 1. *Let C be a symmetric cardinality constraint with cost and $N(C)$ be a value network of C. The following properties are equivalent.*

- *C is consistent*
- *there exists a flow from s to t in $N(C)$ which satisfies lower and upper bounds of capacities on the arcs in $N(C)$ and which cost is less than or equal to H.*

Proof. Assume that C is consistent, thus $T(C) \neq \emptyset$. Consider $P \in T(C)$. We can build a function f in $N(C)$ such that:

1. $\forall x_i \in X(C), f(x_i, t) = \#(x_i, P)$
2. $\forall x_i \in X(C), f(v_j, x_i) = 1$ if v_j appears in the subset $P(x_i)$, otherwise $f(v_j, x_i) = 0$
3. $\forall v_j \in D(X(C)), f(s, v_j) = \#(v_j, C, P)$
4. $cost(f) = \sum_{i=1}^{|X(C)|} cost(x_i, P) \leq H$

Properties (1) – (3) are the feasibility properties of a symmetric cardinality constraint and are proved in [1]. By these properties, if C is consistent then there exists a feasible flow f in $N(C)$. Recall from Equation (1) that the cost of a flow is a sum of products of costs associated with an arc and an amount of flow on respective arc. Since the cost associated with arcs other then arcs between $x_i \in X$ and $v_j \in D(X(C))$ is equal to 0 and, by Property 2, flow f from any v_j to any x_i is equal to 1 if v_j is in a subset $P(x_i)$ then the cost of such flow f is equal to the sum of cost of all $x_i \in X$ in P. Consequently, if C is consistent then $\sum_{i=1}^{|X(C)|} cost(x_i, P) \leq H$ and $cost(f) \leq H$, which proves Property 4.

On the other hand, assume that there exists a feasible flow f from s to t in $N(C)$ with cost lower than or equal to H. Since f is feasible then $\forall x \in X(C) l_{x_i} \leq \#(x_{C_i}, P) \leq u_{x_i}$, $f(x_i, t) = \#(x_i, P)$ and $\forall v \in D(X(C)) l_{v_j} \leq \#(v_j, C, P) \leq u_{v_j} \wedge f(s, v_j) = \#(v_j, C, P)$, and the set of arcs such that $f(i, j) = 1$ corresponds to the set of edges of value graph of symmetric cardinality constraint with costs, which is proved in [1]. Since the arc cost for any arc (s, i) and (i, t), which includes direct arcs between s and t, is equal to 0 and the flow between any i and j can be at most 1, then $cost(f)$ is equal to the sum of costs of arcs between i and j where $f(i, j) = 1$. Since the set of such arcs corresponds to the set of arcs in value graph of C thus $cost(f) = \sum_{i=1}^{|X(C)|} cost(x_i, P)$. Consequently, if $cost(f) \leq H$ so is $\sum_{i=1}^{|X(C)|} cost(x_i, P)$, which proves Proposition 1. □

This proposition gives a way of computing consistency of symmetric cardinality constraint with costs by computing a feasible flow f in $N(C)$ and checking if $cost(f) \leq H$. Since H is independent of $N(C)$ every time this equality does not hold we would need to verify if there exists an other feasible flow f' in $N(C)$ such that $cost(f') < cost(f)$ and $cost(f') \leq H$.

On the other hand, if a feasible flow f in $N(C)$ is a minimum cost flow we can verify that C is consistent if $cost(f) \leq H$, otherwise C is inconsistent.

5 Minimum Cost Flow

There exists several methods of obtaining minimum cost flow in a network. A survey of such methods is given in [5]. Here we describe a simple variant of successive shortest path algorithm from [5].

Given a value network for a symmetric cardinality constraint with costs and a flow f, we can build the residual graph $R(f)$ in the same way as for the symmetric cardinality constraint and extend it with arc costs by

– associating with each arc (i, j) with free capacity its residual cost $rc(i, j) = cost(i, j)$
– associating with each arc (j, i) representing a flow which exceeds the lower bound of (i, j) a residual cost $rc(j, i) = -cost(i, j)$.

For any path p the residual cost of p, denoted $rc(p)$ is a sum of residual costs of all arcs $(i, j) \in p$.

Algorithm 1 Minimum Cost Flow

1: Start with the zero flow f.
2: Pick an arc (i, j) such that $f(i, j)$ violates the lower bound for the flow from i to j.
3: Find p a shortest path from j to i in $R(f) - \{(i, j)\}$.
4: Obtain a new flow f' from f by sending a flow along p and set $f = f'$
5: Goto 2

The algorithm, listed here as Algorithm 1, computes a minimum cost flow in N by repeatedly choosing an arc (i, j) with a flow, which violates the lower bound constraint, and computes a shortest path p from j to i with respect the costs of an arc. After a path p is found a new flow f' is obtained by sending a flow along p. If, at some point, there is no path for the current flow then there is no feasible flow in N. Otherwise, obtained flow is feasible and is a minimum cost flow.

The complexity of the algorithm is dependent on the complexity of implemented shortest path algorithm. However, the most powerful shortest path algorithms require that the costs on all arcs in a graph are non-negative.

The standard method for transforming all arcs costs to non-negative values is to use costs relative to "imputed" costs associated with incident nodes of an arc. These costs are usually referred to as *reduced costs* and the costs associated with incident nodes are referred to as *node potentials*. More formally,

Definition 6.

– *a potential function is a function π which associates with each node $i \in N$ a number $\pi(i)$, which is referred to as a node potential*
– *with respect to the node potentials, the reduced cost c^{π}_{ij} of an arc (i, j) in $R(f)$ is defined by $c^{\pi}_{ij} = rc(ij) - \pi(i) + \pi(j)$*

Given a minimum cost flow f the potential function $\pi^f(i) = d^f_j(i)$, where $d^f_j(i)$ represents the shortest path distance in $R(f)$ from node j to every node $i \in N$. Starting with the zero flow,f^0, all $\pi^{f^0}(i) = 0$ and consequently all $c^{\pi}_{ij} = rc(i, j)$. After each iteration of the algorithm, which computes shortest paths from given node j to every other node in the graph, the potential of each node i is updated with $\pi'(i) = \pi(i) - d(i)$. Then, the cost of each arc in the residual graph can be converted to non-negative reduce cost as in Definition 6. For justification of this method see [5].

The fastest algorithm for computing shortest path in a graph with non-negative arc lengths is Dijkstra algorithm implemented with Fibonacci heaps (see [12, 5] for comparison). The algorithm requires $O(m + n\ log\ n)$ time, where m is the number of arcs and n is the number of nodes in a graph. The number of iterations is bounded by the number of lower bounds on the flow on each arc, which is $\sum l(i,j)$. Thus, the complexity of the shortest paths algorithm can be bounded by $O(\sum l(i,j) \times (m + n\ log\ n))$ time.

In terms of the symmetric cardinality constraint $n = |X(C)| + |D(X(C))| + 2$ and $m = \sum_{i=1}^{|X(C)|} D(x_i) + |X(C)| + |D(X(C))| + 2$. The number of iterations is bounded by $\sum_{i=1}^{|X(C)|} l_{x_i} + \sum_{j=1}^{|D(X(C))|} l_{v_j}$.

6 Filtering of Symmetric Cardinality Constraint with Costs

By Proposition 1, a symmetric cardinality constraint with costs C is consistent if and only if there is a feasible flow in the value network of C, $N(C)$, which cost is less than or equal to H. Thus, if there is a feasible flow in $N(C)$ with an overall cost less than or equal to H which contains a flow along (v_i, x_i) then the value v_i of x_i is consistent with C. Assuming that C is a consistent symmetric cardinality constraint with costs and f is a minimum cost flow in $N(C)$ we can formulate the following proposition.

Proposition 2. *A value v_j of a variable x_i is not consistent with C if and only if*

1. *for all feasible flows f in $N(C)$ $f(v_j, x_i) = 0$*
2. *$d_{R(f)-\{(x_i,v_j)\}}(x_i, v_j) > H - cost(f) - rc(v_j, x_i)$*

Proof. Property 1 states that the value v_j of the variable x_i is inconsistent with C if there is no feasible flow in $N(C)$ containing (v_j, x_i) which is proved in Proposition 1

Furthermore, it is proved (see [8], p. 130) that if there exists a path p from node j to node i in $R(f) - \{(j,i)\}$ then the flow f' obtained from f by sending k units of flow along p has a cost $cost(f') = cost(f) + k(rc(i,j) + d_{R(f)-\{(j,i)\}}(j,i))$. Since the maximum amount of flow which can be pushed through any (i,j) is equal to 1 we obtain $cost(f') = cost(f) + rc(i,j) + d_{R(f)-\{(j,i)\}}(j,i)$. By Proposition 1 C is consistent if cost of a feasible flow is lower than or equal to H, thus if $cost(f') > H$ than C is inconsistent. By substituting $cost(f')$ we obtain $cost(f) + rc(i,j) + d_{R(f)-\{(j,i)\}}(j,i) > H$, which gives $d_{R(f)-\{(x_i,v_i)\}}(x_i, v_j) > H - cost(f) - rc(v_j, x_i)$. This proves the second property. □

By Proposition 6, given an arc (v_j, x_i) such that $f_{v_j x_i} = 0$, if there exists a path from x_i to v_j in $R(f) - \{(x_i, v_j)\}$ then there exists a feasible flow in $N(C)$ containing (v_j, x_i). Moreover, if $d_{R(f)-\{(x_i,v_i)\}}(x_i, v_i) \leq H - cost(f) - rc(v_i, x_i)$ then such a flow has a cost less than or equal to H. Thus, a value v_j of a variable x_i is consistent with C.

Note that due to a special structure of the residual graph obtained from a flow in $N(C)$, if $f(v_j x_i) = 0$ then $R(f)$ does not contain (x_i, v_j). Thus, no modification of $R(f)$ is required in order to perform this computation. The special structure of the residual graph will also insure that a simple path from x_i to v_j will contain at least tree nodes.

Given a minimum cost flow f in a value network of a symmetric cardinality constraint $N(C)$, the consistency of each value $v_j \in D(x_i)$ is verified in $O(|X(C)| \times (m + n\ log\ n))$ time, which is the same as for filtering the global cardinality constraint with costs ([2, 4]).

7 Conclusion

In this paper we introduce the symmetric cardinality constraint with cost. Moreover, we present methods for computing its consistency and methods for filtering domains of its variables. We show that the time complexity for verifying consistency of a symmetric cardinality constraint with costs and for filtering domains of constraint variables is the same as for the global cardinality constraint with costs.

Other variants of the symmetric cardinality constraint with cost, like those constraining the cost associated with the values assigned to individual variables etc., can be easily derived from the constraint described here.

Acknowledgment

References

[1] Kocjan, W., Kreuger, P.: Filtering Methods for Symmetric Cardinality Constraint. Proceedings of 1th Int. Conf. CPAIOR (2004)

[2] Régin, J.-Ch.: Generalized Arc Consistency for Global Cardinality Constraint. Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAI-96) (1996)

[3] Régin, J.-Ch.: Arc Consistency for Global Cardinality Constraints with Costs. Principles and Practice of Constraint Programming — CP'99, Conference Proceedings, (1999) 390–404

[4] Régin, J.-Ch.: Cost–Based Arc Consistency for Global Cardinality Constraints. Constraints, **7** (2002) 387–405

[5] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows. Theory, algorithms and applications. Prentice–Hall Inc., (1993)

[6] Tsang, E.: Foundation of Constraint Satisfaction Academic Press (1993)

[7] Laurière, J.-L.: A language and a program for stating and solving combinatorial problems. Artificial Intelligence, **10** (1978) 29–127

[8] Lawler, E.: Combinatorial Optimization: Network and Matroids, Holt, Rinehart and Winston (1976)

[9] Ahuja, R.K., Kodialam, A., Mishra, A.K., Orlin, J.B.: Computational Investigations of Maximum Flow Algorithms. European Journal of Operational Research, **97** (1997) 509–542

[10] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. Second Edition, The MIT Press (2001)
[11] Tarjan, R.E.: Depth–First Search and Linear Graph Algorithms. SIAM J. Computing, **1** (1972) 146–160
[12] Cherkassky, B.V., Goldberg, A.V., Radzik, T.: Shortest Paths Algorithms: Theory and Experimental Evaluation. Mathematical Programming **73 (1996) 129–174**