A Circuit Constraint for Multiple Tours Problems

Philippe Vismara, Nicolas Briot

A Circuit Constraint for Multiple Tours Problems

Philippe Vismara[1,2] and Nicolas Briot[1]

[1] LIRMM, Univ Montpellier, CNRS, Montpellier, France
[2] MISTEA, Montpellier SupAgro, INRA, Univ Montpellier, Montpellier, France
philippe.vismara@supagro.fr

This PDF file is a pre-print version of the final publication which can be found at:
https://doi.org/10.1007/978-3-319-98334-9_26

Abstract. Routing problems appear in many practical applications. In the context of Constraint Programming, circuit constraints have been successfully developed to handle problems like the well-known Traveling Salesman Problem or the Vehicle Routing Problem. These kind of constraints are linked to the search for a Hamiltonian circuit in a graph. In this paper we consider a more general multiple tour problem that consists in covering a part of the graph with a set of minimal cost circuits. We define a new global constraint WeightedSubCircuits that generalizes the WeightedCircuit constraint by releasing the need to obtain a Hamiltonian circuit. It enforces multiple disjoint circuits of bounded total cost to partially cover a weighted graph, the subsets of vertices to be covered being induced by external constraints. We show that enforcing Bounds Consistency for WeightedSubCircuits is NP-hard. We propose an incomplete but polynomial filtering method based on the search for a lower bound of a weighted Steiner circuit.

1 Introduction

Many real problems can be modeled as a tour problem, the best known is the Travelling Salesman Problem (TSP). It consists in finding a Hamiltonian cycle (i.e., passing by each vertex of a graph once) with a minimum weight. Many works coming from Integer Linear Programming (ILP) or dynamic programming allow to quickly solve large instances of TSP. In addition, some variations of the TSP, such as the Vehicle Routing problem (VRP), have been the subject of numerous studies proposing effective solving methods [18].

In this context, Constraint Programming has for a long time offered its expressiveness to address variations of TSP. Initially limited to small instances, the most recent filtering algorithms allow to compete with ILP approaches on complex problems where complementary constraints restrict TSP solutions.

These good results are related to the definition of global constraints and the associated filtering algorithms: the constraint Cycle (or Circuit) enforces covering the graph with one circuit visiting all vertices once; the constraint

WeightedCircuit imposes, in addition, that the sum of the costs of the edges of the circuit is lower than a cost variable. As enforcing arc consistency for these constraints is generally NP-hard, the filterings used are inevitably incomplete but in practice relatively efficient. For instance, the WeightedCircuit constraint propagator can incorporate different methods based on TSP relaxation from literature.

However, all these constraints are linked to the search for a Hamiltonian cycle. Many real problems require searching for one or more cycles covering all or part of the vertices of the graph. These problems correspond to two kinds of relaxation in the definition of the Hamiltonian cycle. The classical relaxation is the VRP where some vertices (depots) can be visited several times. This problem is generally modeled by duplicating a few vertices in order to return to a Hamiltonian cycle. The second relaxation consists in not covering all the vertices of the graph, the set of the discarded vertices depending on external constraints. In this case, the Cycle constraint can eventually be used by artificially adding the discarded vertices to the end of the Hamiltonian cycle. However, it becomes impossible to integrate them into a weighted cycle without disrupting the cost of the solution. It is then impossible to benefit from all the filtering power of constraint WeightedCircuit.

In this paper, we aim to generalize the WeightedCircuit constraint in case some vertices can be discarded. We consider a more general multiple tour problem that consists in covering a part of the graph by a set of minimal cost circuits. We define a new global constraint, called WeightedSubCircuits, that enforces multiple disjoint circuits of bounded cost to partially cover a weighted graph. This is a generalization of constraint WeightedCircuit where the Hamiltonian circuit can be divided into several disjointed subcircuits, with an additional subset of discarded vertices.

The remainder of this paper is structured as follows. Section 2 surveys the necessary preliminaries. Section 3 covers related work on global constraints for tour problems. Section 4 gives the definition of WeightedSubCircuits constraint and proposes a decomposition of the constraint with standard constraints and a cycle constraint adapted to multiple tours. Section 5 deals with the filtering of this NoSubTours constraint and that of the WeightedSubCircuits constraint. Finally, Section 6 presents some preliminary experimental results.

2 Preliminaries

We consider a weighted graph $G = (V, E, c)$ where V is a set of vertices, E a set of edges and a weight function $c : E \to \mathbb{Q}_+$.

When the graph G respects *triangle inequality*, the weight of any edge (i, j) is smaller or equal to the cost of any path from i to j.

Graphs are considered from an oriented point of view. In this context, the term circuit should be used rather than cycle. However, a cycle is generally described by a sequence of vertices or by defining, for each vertex i, the $Next_i$

vertex that follows i in the cycle. In both cases, the cycle is oriented and then it is a circuit.

An *elementary* circuit of G is a circuit where no vertex appears more than once. A *Hamiltonian* circuit is an elementary circuit of length $|V|$.

For any subset $W \subseteq V$, $G[W]$ is the subgraph of G induced by W.

For any set variable Set, the lower (respectively upper) bound of Set, denoted by $\text{LB}(Set)$ (respectively $\text{UB}(Set)$), is the set of required (respectively possible) values in Set.

In the following, we will note $OPT_{TSP}(G)$ the cost of an optimal solution for $TSP(G)$.

3 Related work

The Constraint Programming research community has long been interested in the search for Hamiltonian circuits, which were already part of the Alice language constraints [12]. A configurable Cycle constraint was also part of the global constraints introduced in the CHIP solver [1].

The most common model is to define a variable $Next_i$ for each vertex i of the graph, where $Next_i$ represents the vertex which follows i in the Hamiltonian circuit. To ensure that each vertex is visited only once, an AllDifferent constraint can be posted on the $Next_i$ and enforcing GAC for this constraint is polynomial [17]. Conversely, checking that there is a Hamiltonian circuit is NP-complete. The filtering used in solvers for the Circuit constraint is, therefore, naturally incomplete. Two subconstraints are mainly used for this filtering on the edges composing the circuit: the NoSubTour constraint which prohibits the presence of subcircuits and the Connected constraint which ensures the strong connectivity of the circuit. Other works have proposed a filtering based on graph separators [11] or investigated how to add explanations to the Circuit constraint in a lazy clause generation solver [9].

The NoSubTour constraint [3,14] is posted on graph $G = (V, E)$. It ensures that the $Next_i$ variables do not form a subtour of length strictly smaller than $|V|$. Combined with constraint AllDifferent, the NoSubTour constraint enforces $Next_i$ variables to form a Hamiltonian circuit of G.

The filtering generally associated with the NoSubTour constraint consists in removing from $Next_i$ any value that could close a path to form a subcircuit strictly smaller than $|V|$. This is based on the following rule (Figure 1) applied when $Next_i$ is instantiated with j:

$$Next_i = j \ \wedge \ (\mathcal{L}(\beta(i)) + \mathcal{L}(j) + 1) < |V| - 1 \ \Rightarrow \ Next_{\varepsilon(j)} \neq \beta(i) \qquad (1.1)$$

where, for any vertex z, $\beta(z)$, $\varepsilon(z)$ and $\mathcal{L}(z)$ are respectively the begining, the end and the length of the path induced by the $Next_i$ variables and passing through z. Due to constraint AllDifferent, we must have $\varepsilon(i) = i$ and $\beta(j) = j$ when $Next_i$ is instantiated with j.

The values of $\beta(z)$, $\varepsilon(z)$ and $\mathcal{L}(z)$ can be easily managed with backtrackable variables and updated in $O(1)$ for each instantiation of $Next_z$.

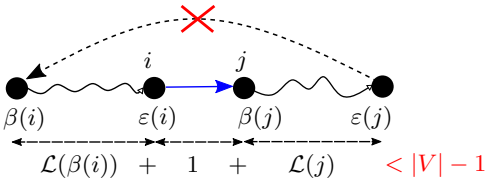


Fig. 1. Illustration of the filtering rule for the NoSubTour constraint.

The CONNECTED constraint has been less studied in literature. The simplest approach is to use a $O(|V|+|E|)$ search algorithm (like Tarjan's) to find strong connected components [6]. It is also possible to limit the number of searches by maintaining a spanning tree [15].

Whatever the filtering used for the CIRCUIT constraint, it is only concerned with the connectivity of the graph without taking into account the edge weights. However, the cost of the circuit is a key element in the optimization of a TSP or VRP. Therefore, constraint WEIGHTEDCIRCUIT($Next_1, \ldots, Next_n, Cost$) has been defined to enforce the rule that the circuit defined by the variables $Next_i$ has a lower cost than the variable $Cost$.

The filtering algorithms used for constraint WEIGHTEDCIRCUIT are based on TSP relaxations. This consists in reducing the TSP to a polynomial optimization problem whose optimal cost is a lower bound of the initial TSP cost. This bound can be used directly to update LB($Cost$). It also filters the $Next_i$ variables by eliminating the edges which are not part of the optimal solution and which, if they replace an edge of this solution, generate an additional cost beyond UB($Cost$).

Several relaxations can be used to filter the constraint WEIGHTEDCIRCUIT [3,14,8,2] and most of them are incomparable [7]. The Minimum Spanning Tree (MST) and even better the Minimum Spanning Arborescence (MSA) directly provide a lower bound to the TSP. In Assignment Problem (AP) relaxation, the solution can be composed of several disjointed cycles. With Held and Karp 1-tree relaxation, the solution is a Minimum Spanning Tree of $G[V \setminus \{1\}]$ plus two edges connecting vertex 1 to this spanning tree.

4 The WeightedSubCircuits Constraint

The WEIGHTEDSUBCIRCUITS (WSC) constraint aims to generalize the WEIGHTEDCIRCUIT constraint. Instead of imposing a single Hamiltonian circuit on the whole graph, it enforces a Hamiltonian circuit for each subgraph induced by one or more subsets of vertices.

For the sake of simplicity, we assume that these subsets are defined by $K+1$ set variables Set_1,...,Set_K, Set_{dummy}, the last subset being the set of discarded vertices. However, it is possible to adapt the definition with other representations of these subsets, for example with integer variables.

The WeightedSubCircuits constraint is intended to be combined with other constraints controlling the distribution of vertices in subsets Set_1,..., Set_K, Set_{dummy} which must form a partition of the set of vertices. The number K of subsets is an upper bound as some subsets may be empty. In addition, non-empty subsets must contain at least 2 vertices since the isolated vertices must belong to Set_{dummy}.

4.1 Definition

Definition 1. (WeightedSubCircuits)
Given a weighted graph $G = (V, E, c)$, the constraint

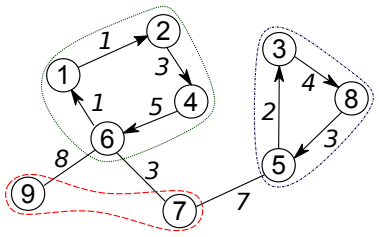
$$\text{WSC}[G]([Set_1, ..., Set_K], Set_{dummy}, [Next_1, ..., Next_n], [Cost_1, ..., Cost_K], Z)$$

holds on the set variables $Set_1, ...Set_K, Set_{dummy}$ and the integer variables $Next_1$, ..., $Next_n$, Z and $Cost_1, ..., Cost_K$ if and only if:

1. *the subsets $Set_1, \ldots, Set_K, Set_{dummy}$ form a partition of V;*
2. *$\forall k \in 1..K$, the set $E_k = \{(i, Next_i) \mid i \in Set_k\}$ defines a Hamiltonian circuit of $G[Set_k]$ and $\sum_{(i,j)\in E_k} c(i,j) \leq Cost_k$;*
3. *$\forall i \in V$, $i \in Set_{dummy} \Leftrightarrow Next_i = i$;*
4. *$\sum_{k=1}^{K} Cost_k \leq Z$;*

This definition is illustrated by Figure 2. The set Set_{dummy} contains all discarded vertices. As in Minizinc [13] `subcircuit` constraint, we set $Next_i = i$ for all discarded vertices. This allows constraint AllDifferent to be applied to all $Next_i$ variables.

Because of the third rule, any subset Set_k must be empty or contains at least 2 vertices. When $K = 1$, the constraint can be simplified since $Set_{dummy} = V \backslash Set_1$ and Z is an upper bound of $Cost_1$.



$Set_1 = \{1, 2, 6, 4\}$
$Set_2 = \{3, 5, 8\}$
$Set_{dummy} = \{7, 9\}$
$Cost_1 = 10 \quad Cost_2 = 9$
$Z = 20$
$Next_1 = 2 \quad Next_2 = 4 \quad Next_3 = 8$
$Next_4 = 6 \quad Next_5 = 3 \quad Next_6 = 1$
$Next_7 = 7 \quad Next_8 = 5 \quad Next_9 = 9$

Fig. 2. An example of a weighted graph and a set of variables for which the constraint $\text{WSC}[G]([Set_1, Set_2], Set_{dummy}, [Next_1, ..., Next_9], [Cost_1, Cost_2], Z)$ holds.

In the definition of the WeightedSubCircuits constraint we have assumed that the subsets are represented by set variables. This can facilitate the addition of external constraints on the subsets, such as a maximum cardinality

to limit the number of vertices in each circuit. However, the constraint could also be defined with subsets represented by n membership integer variables $\{member_i\}_{i \in V}$. Moreover, the two representations can be combined thanks to channeling constraints $\forall i \in V, \forall k \in 1..K,\ i \in Set_k \Leftrightarrow member_i = k$ and $i \in Set_{dummy} \Leftrightarrow member_i = K + 1$.

The WeightedSubCircuits constraint is clearly a generalization of the WeightedCircuit constraint:

$$\text{WeightedCircuit}[G](Next_1, ..., Next_n, Z) \equiv \\ \text{WSC}[G]([V], \emptyset, [Next_1, ..., Next_n], [Z], Z)$$

Moreover, the WeightedSubCircuits constraint can also be used to implement some (but not all) variants of the generic Cycle constraint of CHIP [1]. For instance, one variant of the CHIP Cycle constraint holds for K cycles with p incompatible nodes ($p \leq K$) that must belong to disjoint cycles with total cost constrained by two bounds. This cycle constraint is equivalent to a WeightedSubCircuits constraint with additional unary constraints on the $Cost_k$ and Set_k variables.

Since WeightedSubCircuits is a generalisation of WeightedCircuit, it is not surprising that filtering WeightedSubCircuits is NP-hard:

Theorem 1. *Achieving Bounds Consistency (BC) on* WSC *is NP-hard.*

Proof. Deciding if a graph $G = (V, E, c)$ has a Hamiltonian circuit of cost less or equal to a given value p is NP-complete. Consider the constraint

$$\text{WSC}[G]([Set_1], Set_{dummy}, [Next_1, ..., Next_n], [Cost_1], Z)$$

where $\text{LB}(Set_1) = V$ and for each variable $Next_i$, $dom(Next_i) = \{j \mid (i, j) \in E\}$ and $D(Cost_1) = D(Z) = \{0, ..., p\}$. BC empties the domain of Z if and only if G does not admit a Hamiltonian circuit of cost less or equal than p □

4.2 Decomposition

Before considering the development of specific propagators, we can try to decompose the WSC constraint into a set of standard constraints. Except for line 1.6, the WSC constraint can be decomposed into standard constraints as follows:

Proposition 1.

$$\text{WSC}[G](\ Set_1, ...Set_K, Set_{dummy}, Next_1, ..., Next_n, Cost_1, ..., Cost_K, Z) \Leftrightarrow$$

$$\text{AllDifferent}(Next_1, ..., Next_n) \quad (1.2)$$

$$\wedge \quad \text{Partition}(Set_1, ..., Set_K, Set_{dummy}) \quad (1.3)$$

$$\wedge \quad \forall i \in V,\ i \in Set_{dummy} \Leftrightarrow Next_i = i \quad (1.4)$$

$$\wedge \quad \forall i \in V,\ \forall k = 1..K,\ i \in Set_k \Leftrightarrow Next_i \in Set_k \quad (1.5)$$

$$\wedge \quad \text{NoSubTours}(Set_1, ..., Set_K, Next_1, ..., Next_n) \quad (1.6)$$

$$\wedge \quad \forall k = 1..K,\ \textstyle\sum_{i \in Set_k} c(i, Next_i) \leq Cost_k \quad (1.7)$$

$$\wedge \quad \textstyle\sum_{k=1}^{K} Cost_k \leq Z \quad (1.8)$$

The AllDifferent constraint on the $Next$ variables ensures that any vertex must belong to a cycle or must be isolated and then, thanks to line 1.4, must belong to Set_{dummy}. We assume that the Partition constraints allows empty sets. Thanks to line 1.5 each cycle must be included in a single subset Set_k. The NoSubTours constraint enforces that such a cycle is an Hamiltonian cycle of the induced subgraph $G[Set_k]$. This is not a standard constraint but it is a generalization of the NoSubTour constraint[3,14]. The next section will discuss NoSubTours filtering. The sum constraints (1.7) and (1.8) ensure that the cost of each cycle is greater or equal to the sum of the weights of its edges and that the total sum of $Cost_k$ variables is less or equal to Z.

5 Propagation

First we will look at the filtering of constraint NoSubTours in order to be able to implement constraint WeightedSubCircuits thanks to its decomposition. We will then propose a specific additional filtering for constraint WeightedSubCircuits.

Since obtaining AC for these two constraints is NP-hard, the filtering algorithms that we will study in this section are necessarily incomplete.

5.1 NoSubTours

When $Next_i$ is instantiated with j, the filtering rule 1.1 used for the constraint NoSubTour is dedicated to remove from the domain of $Next_{\varepsilon(j)}$ any value leading to a cycle of size less that $|V|$.

For the NoSubTours constraints (1.6), the set $\{(i, Next_i), i \in Set_k\}$ must form a cycle of size $|Set_k|$ in $G[Set_k]$. Hence, if the path resulting from the instantiation $Next_i = j$ has a length smaller than the size of the lower bound of Set_k, this path cannot be closed at its ends to form a cycle. This corresponds to the following filtering rule:

$$Next_i = j \ \wedge \ i \in \text{LB}(Set_k) \ \wedge \ (\mathcal{L}(\beta(i)) + \mathcal{L}(j) + 1) < |\text{LB}(Set_k)| - 1 \\ \Rightarrow \ Next_{\varepsilon(j)} \neq \beta(i) \quad (1.9)$$

Conversely, if the resulting path passes through all vertices of the upper bound of Set_k, the cycle must be closed and Set_k is instantiated:

$$Next_i = j \ \wedge \ i \in \text{LB}(Set_k) \ \wedge \ (\mathcal{L}(\beta(i)) + \mathcal{L}(j) + 1) = |\text{UB}(Set_k)| - 1 \\ \Rightarrow \ Next_{\varepsilon(j)} = \beta(i) \ \wedge \ Set_k = \text{UB}(Set_k) \quad (1.10)$$

NoSubTours filtering can also benefit from searching for connected components to ensure that $\text{LB}(Set_k)$ is included in a connected (via $Next_i$ variables) component of $\text{UB}(Set_k)$.

Finally, we can notice that backtractable variables like $\beta(i)$ can also be used to filter constraint (1.5) since all the vertices in the path passing through i and

connecting $\beta(i)$ to $\varepsilon(i)$ must be in the same Set_k than i. This corresponds to the following filtering rules:

$$i \in \text{LB}(Set_k) \Rightarrow \{\beta(i), Next_{\beta(i)}, \ldots, i, Next_i, \ldots, \varepsilon(i)\} \subseteq \text{LB}(Set_k) \quad (1.11)$$

and

$$i \notin \text{UB}(Set_k) \Rightarrow \{\beta(i), Next_{\beta(i)}, \ldots, i, Next_i, \ldots, \varepsilon(i)\} \cap \text{UB}(Set_k) = \emptyset \quad (1.12)$$

5.2 WeightedSubCircuits

Previous works on the WeightedCircuit constraint have shown the benefit of dedicated filtering compared to separate filtering on the Circuit constraint and the cost of the circuit.

During the search, the vertices involved in the WeightedSubCircuits constraint can be divided in 4 categories (see Figure 3) :

A. The vertices in $\text{LB}(Set_{dummy})$ will not be part of the circuits.
B. The vertices in $\text{LB}(Set_k)$ will necessarily contribute to the value of $Cost_k$ and Z.
C. The vertices in $V \setminus (\text{UB}(Set_{dummy}) \cup \bigcup_k \text{LB}(Set_k))$ cannot be excluded but are not yet assigned to a Set_k. They cannot yet contribute to the value of a specific $Cost_k$ but will necessarily contribute to the value of Z.
D. For the vertices in $\text{UB}(Set_{dummy}) \setminus \text{LB}(Set_{dummy})$ it is still too early to know if they will be part of the circuits.

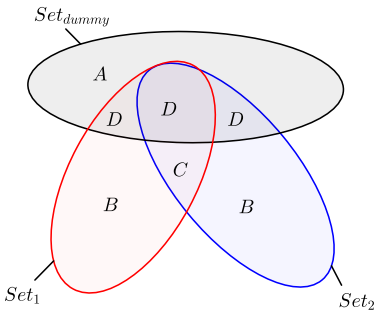


Fig. 3. Vertex distribution for constraint WeightedSubCircuits during the search: A vertices are definitely discarted and inserted in subset Set_{dummy}; B vertices are definitely added to one subset Set_k; C vertices can no longer be discarded but are not yet assigned to any subset S_k; D vertices can belong to any subset.

Whether one considers a particular variable $Cost_k$ or the global variable Z, in both cases there is a subset of vertices that must be part of the solution and other vertices that may participate.

For instance, to find a lower bound for $\text{LB}(Cost_k)$, the subset of required vertices is equal to $\text{LB}(Set_k)$. It is unfortunate that it is not possible to consider only $G[\text{LB}(Set_k)]$ in order to find a lower bound for $\text{LB}(Cost_k)$:

Proposition 2. *$OPT_{TSP}(G[\text{LB}(Set_k)])$ is not a lower bound for $\text{LB}(Cost_k)$*

Proof. In the graph of Figure 4, the subgraph induced by $\text{LB}(Set_k)$ is a cycle of weight 24. With additional vertices 5 and 6 added to Set_k, the induced subgraph includes a cycle of weight 21. Moreover, $G[\text{LB}(Set_k)]$ may not contain a cycle while $G[\text{UB}(Set_k)]$ does □

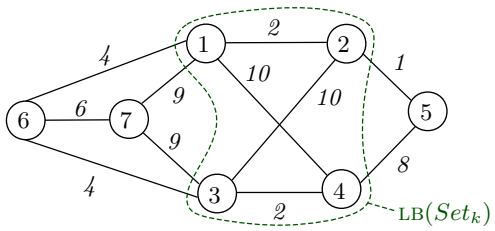


Fig. 4. The optimal value for $TSP(G[\text{LB}(Set_k)])$ is equal to 24 whereas $OPT_{TSP}(G[\text{LB}(Set_k) \cup \{5, 6\}])$ is equal to 21.

To find a lower bound for $Cost_k$, we must consider not only the mandatory vertices but also the potential vertices. This question can be reduced to the Steiner cycle problem, which is a generalization of the Steiner tree problem [10].

Definition 2. *(Steiner Cycle Problem) Let $H = (V, E, c)$ a weighted graph and $V' \subseteq V$. The* Steiner Cycle Problem *$SCP(H, V')$ consists in finding an elementary cycle of minimum cost that contains all nodes in V' (but may include additional vertices). The cost of an optimal solution will be noted $OPT_{SCP}(H, V')$.*

The TSP is a specific case of SCP where $V' = V$.

Proposition 3. $\text{LB}(Cost_k) \geq OPT_{SCP}(G[\text{UB}(Set_k)], \text{LB}(Set_k))$

Proof. $Cost_k$ is the cost of a Hamiltonian cycle in $G[Set_k]$. Thus, this cycle must necessarily pass through all the vertices of $\text{LB}(Set_k)$ and eventually through some vertices of $\text{UB}(Set_k) \setminus \text{LB}(Set_k)$. This is the exact definition of $SCP(G[\text{UB}(Set_k)], \text{LB}(Set_k))$ □

Proposition 3 gives a way to filter $Cost_k$. Since computing an optimal Steiner cycle is NP-hard, a lower bound can be determined by relaxing a constraint of SCP as is done for TSP. To do this, we start by defining an extended subgraph:

Definition 3. *(Extended subgraph) Given a weighted graph $H = (V, E, c)$ and a subset $V' \subseteq V$, the* extended subgraph $\mathcal{G}(H, V')$ *is obtained by adding to $H[V']$ new edges (i, j), with weight $c(i, j) = \delta_{i,j}$, such that $(i, j) \notin E$ and there is a shortest path connecting i to j in $H[(V \setminus V') \cup \{i, j\}]$ whose cost is equal to $\delta_{i,j}$.*

This definition is illustrated by Figure 5. The edges added to $H[V']$ correspond to a shortest path outside V' and connecting two non-adjacent vertices of $H[V']$.

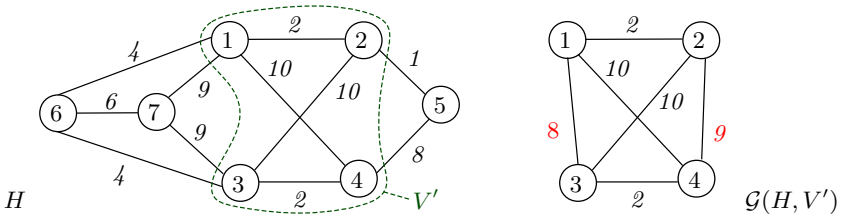


Fig. 5. Extended subgraph $\mathcal{G}(H, V')$ for a graph $H = (V, E, c)$ and a subset $V' \subseteq V$

Proposition 4. *Given a weighted graph $H = (V, E, c)$ that respects triangular inequality and $V' \subseteq V$ with $|V'| \geq 3$ we have*

$$OPT_{SCP}(H, V') \geq OPT_{TSP}(\mathcal{G}(H, V'))$$

Proof. Any Hamiltonian cycle $\mathcal{C}$ that is solution of $SCP(H, V')$ is composed of paths included in V' and paths outside V'. Let $\mathcal{P}_{i,j} = \langle i, x_1, \ldots, x_t, j \rangle$ be a sub-path of $\mathcal{C}$ such that $i, j \in V'$ and $\forall p \in 1..t, x_p \notin V'$. If $(i, j) \in E$, the triangular inequality imposes that $c(i, j)$ is not greater than the cost of $\mathcal{P}_{i,j}$. If $(i, j) \notin E$, by construction of $\mathcal{G}(H, V')$, there exists in $\mathcal{G}(H, V')$ an edge (i, j) whose weight is lower or equal to the cost of $\mathcal{P}_{i,j}$. Thus, the cycle obtained by replacing in $\mathcal{C}$ all paths $\mathcal{P}_{i,j}$ by edge (i, j) is a Hamiltonian cycle of $\mathcal{G}(H, V')$ whose cost is lower or equal to that of $\mathcal{C}$ □

By combining propositions 3 and 4 we obtain a filtering rule for $Cost_k$:

Corollary 1. $\text{LB}(Cost_k) \geq OPT_{TSP}(\,\mathcal{G}(G[\text{UB}(Set_k)], \text{LB}(Set_k))\,)$

and therefore, a filtering rule for Z:

Corollary 2. $\text{LB}(Z) \geq \sum_{k=1}^{K} OPT_{TSP}(\,\mathcal{G}(G[\text{UB}(Set_k)], \text{LB}(Set_k))\,)$

Since all Set_k must be disjoined, all graphs $\mathcal{G}(G[\text{UB}(Set_k)], \text{LB}(Set_k))$ are disjoined and so we have:

Proposition 5. $\text{LB}(Z) \geq OPT_{TSP}(\,\bigcup_{k=1}^{K} \mathcal{G}(G[\text{UB}(Set_k)], \text{LB}(Set_k))\,)$

Computing $\cup_{k=1}^{K} \mathcal{G}(G[\text{UB}(Set_k)], \text{LB}(Set_k))$ can be done in $O(|V|(|E|+|V|\log|V|))$ with at most $|V \setminus \text{UB}(Set_{dummy})|$ calls to Dijkstra's algorithm. This is comparable to the complexity of some relaxation algorithms. For example, the Hungarian algorithm used for the Assignment Problem relaxation is in $O(|V|^3)$.

Thanks to Proposition 5, a lower bound of Z can be computed with a relaxation of the TSP, as in the case of constraint WeightedCircuit.

This bound is directly related to the state of variables Set_k, which define the vertices of the extended subgraphs, and to the domains of variables $Next_i$, which fix adjacency in G. Depending on the relaxation used, it is also possible to take into account variable Set_{dummy}.

For example, suppose we use the relaxation corresponding to the Assignment Problem (AP). For any graph H, a solution of $AP(H)$ is a set of disjoined minimum cost elementary circuits covering all the vertices of H. Applying AP to extended subgraphs results in a set of sub-cycles covering all type B vertices of Figure 3. However, type C vertices are not covered by these sub-cycles even though they will necessarily be part of the final cycles. To take into account these vertices we can expand the extended subgraphs to vertices of type C.

Let $\mathcal{S}_B = \bigcup_{k=1}^{K} \text{LB}(Set_k)$ the set of type B vertices and $\mathcal{S}_C$ the set of type C vertices. We have $\mathcal{S}_C = V \setminus (\text{UB}(Set_{dummy}) \cup \mathcal{S}_B)$.

We define the *global extended subgraph* $\mathcal{G}^*$ as follow:

Definition 4. *(Global extended subgraph) The* global extended subgraph $\mathcal{G}^*$ *is an extended subgraph built on* $G[\mathcal{S}_B \cup \mathcal{S}_C]$ *by adding only edges between two vertices of the same* $\text{LB}(Set_k)$ *or between a vertex of* $\mathcal{S}_B$ *and a vertex of* $\mathcal{S}_C$*.*

This definition is illustrated by Figure 6.

Computing $\mathcal{G}^*$ has the same complexity $O(|V|(|E|+|V|\log|V|))$ as computing all extended subgraphs.

Since $\mathcal{G}^*$ is built from a subgraph that contains the C-type vertices, we have a new lower bound for Z:

Proposition 6.

$$\text{LB}(Z) \geq OPT_{AP}(\mathcal{G}^*) \tag{1.13}$$

The proof is similar to that of Proposition 4.

For the graph in Figure 6, the cycle set $\{\langle 1,2,3,4,1\rangle, \langle 8,11,8\rangle, \langle 9,10,9\rangle\}$ is an optimal solution to $AP(\mathcal{G}^*)$. The cost (33) of this solution is a lower bound of Z, whatever the vertices added to Set_1 or Set_2. Moreover, according to the upper bound of Z, some edges can be discarded because they cannot be part of a solution. This filtering is based on computing a reduced cost for each arc (i,j) not in the solution of $AP(\mathcal{G}^*)$, i.e., the minimum increase of the overall cost for setting $Next_i$ to j (see [2,7]). For instance, with $\text{UB}(Z) = 40$, edges $(8,10)$ and $(9,11)$ could be eliminated. This filtering concerns only the edges of $\mathcal{G}^*$ that belong to G.

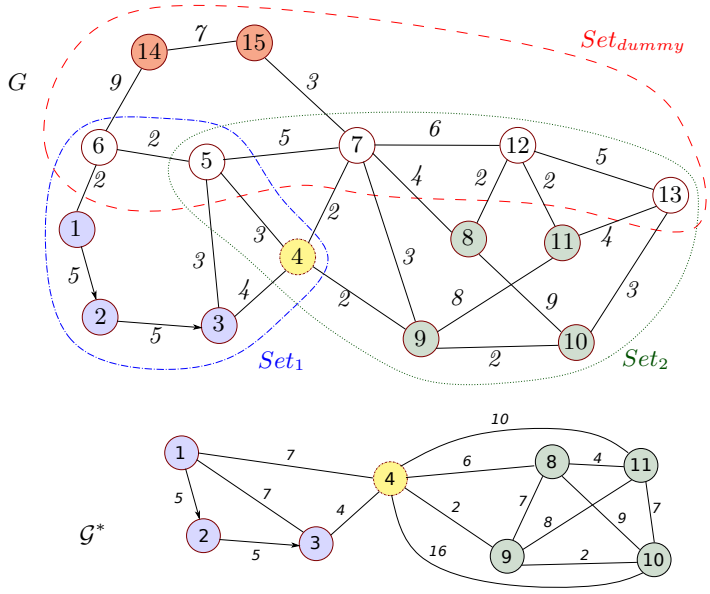


Fig. 6. En example of global extended subgraph $\mathcal{G}^*$, with $\mathcal{S}_C = \{4\}$ and $\mathcal{S}_B = \text{LB}(Set_1) \cup \text{LB}(Set_2) = \{1, 2, 3, 8, 9, 10, 11\}$ and $Next_1 = 2$, $Next_2 = 3$

6 Experimental Results

This section presents some preliminary results to evaluate the benefits of the constraint WeightedSubCircuits. These preliminary experiments aim to measure the interest of a filtering based on Steiner cycles.

Rather than generating random data we consider the Balancing Bike Sharing Systems (BBSS). This problem is linked to the management of a shared bicycle fleet. The objective is to optimize a tour of the stations in order to remove bicycles from overfilled stations and refill empty stations. The capacity of the transport vehicle and the time available do not allow all stations to be optimized. We implemented the CSP model of [4,5] which uses a Cycle constraint. We simply modified it in order to make dummy vertices appear: to ensure that some stations will not be visited, we imposed that the demand for each visited station be fully satisfied rather than partially handle all stations. The benchmark is based on instances from the city of Vienna given by [4]. To gerenate additional instances from size 12 to 18, we extracted a subset of vertices from instances with 20 vertices. Unlike the initial article which was based on a Large Neighborhood Search (LNS) approach, we used a standard search procedure used with fixed ordering of variables. This limits the size of instances that can be processed.

We compared 4 models based on:

– a simple Cycle constraint using the NoSubTour filtering.

- the decomposition of the WeightedSubCircuits constraint presented in section 5.1, with the filtering rules 1.9 to 1.12 for the NoSubTours constraint
- the NoSubTours constraint plus the filtering rule for Z based on $AP(\mathcal{G}^*)$ (Proposition 6)
- the previous filtering rules with additional filtering on the $Next_i$ variables using reduced costs from $AP(\mathcal{G}^*)$

We implemented these models in the Java library Choco 4 [16]. All the experiments were executed on a Linux machine with Intel(R) Xeon(R) CPU E5-2680 (2.40 GHz). The time limit for each run was set to 2 hours.

Table 1 summarizes the results obtained for series of 30 graphs of different sizes. When all the graphs in a series have been resolved, the CPU time and the number of nodes are an average over the 30 graphs. Otherwise, only the number of resolved instances is displayed.

	Circuit	Decomposition of WSC		WSC filtering on Z with $AP(\mathcal{G}^*)$ relaxation		WSC filtering on Z and $Next_i$	
$\|V\|$	# solved	time (sec)	nodes	time (sec)	nodes	time (sec)	nodes
10	12/30	13	201,658	6	78,735	6	66,984
12	6/30	89	1,415,712	16	228,809	12	159,703
14	0/30	381	6,035,049	49	723,136	40	522,914
16		1515	23,729,425	181	2,480,824	121	1,489,637
18		26/30		526	6,480,702	369	4,017,667
20		19/30		1523	16,147,092	1097	10,475,711

Table 1. Average results on BBS instances for the WeightedSubCircuits.

These preliminary results show that the model based on the decomposition of the WeightedSubCircuits constraint allows to find an optimal solution for instances up to 20 vertices while the model based on the Circuit constraint of Choco (NoSubTour) reaches the time limit for several small instances. In addition, the filtering of the Z variable based on the $AP(\mathcal{G}^*)$ relaxation seems to be quite effective. The last columns show that the computation of $\mathcal{G}^*$ is even more profitable if it is used to eliminate edges by filtering the $Next_i$ variables.

7 Conclusion

In this paper, we considered circuit constraints that allow the modeling of tour problems in a CP solver. We have proposed a new global constraint, named WeightedSubCircuits, that enforces multiple disjoint circuits of bounded total cost to partially cover a weighted graph. The constraint is posted on a family of subsets of vertices to obtain a Hamiltonian circuit in each subgraph induced by a subset. The WeightedSubCircuits constraint is intended to be combined with other constraints that control the composition of these subsets and the dummy set of discarded vertices.

We have shown that the WeightedSubCircuits constraint can improve filtering where the WeightedCircuit constraint cannot be used because of the dummy vertices. We have proposed an adaptation of the NoSubTour constraint filtering that is compatible with discarded vertices. We have shown that computing a lower bound of the cost of each circuit can be reduced to a Steiner circuit problem. We demonstrated how to obtain a lower bound of the Steiner circuit by solving a TSP relaxation on an extended subgraph. To obtain a lower bound of the total cost of all circuits, we have shown that it is possible to take into account the required vertices that are not yet assigned to a subset, using AP relaxation. Preliminary experiments have shown the potential of this approach and encourage further exploration of filtering rules for the new constraint.

Acknowledgement This work has been realized with the support of the High Performance Computing Platform HPC@LR, financed by the Languedoc-Roussillon – Midi-Pyrenees Region, Montpellier Mediterranean Metropole and the University of Montpellier.

References

1. Beldiceanu, N., Contejean, E.: Introducing global constraints in chip. Mathl. Comput. Modelling **20**(12), 97–123 (1994)
2. Benchimol, P., Hoeve, W.J.v., Régin, J.C., Rousseau, L.M., Rueher, M.: Improved filtering for weighted circuit constraints. Constraints **17**(3), 205–233 (2012)
3. Caseau, Y., Laburthe, F.: Solving small tsps with constraints. In: Proceedings of the 14th International Conference On Logic Programing. pp. 316–330. MIT Press (1997)
4. Di Gaspero, L., Rendl, A., Urli, T.: Constraint-based approaches for balancing bike sharing systems. In: CP 2013: Principles and Practice of Constraint Programming. Lecture Notes in Computer Science, vol. 8124, pp. 758–773. Springer (2013)
5. Di Gaspero, L., Rendl, A., Urli, T.: Balancing bike sharing systems with constraint programming. Constraints **21**(2), 318–348 (2016)
6. Dooms, G.: The CP(Graph) Computation Domain in Constraint Programming. Ph.D. thesis, Université catholique de Louvain (2006)
7. Ducomman, S., Cambazard, H., Penz, B.: Alternative filtering for the weighted circuit constraint: Comparing lower bounds for the TSP and solving TSPTW. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. pp. 3390–3396 (2016)
8. Focacci, F., Lodi, A., Milano, M.: A hybrid exact algorithm for the TSPTW. INFORMS Journal on Computing **14**(4), 403–417 (2002)
9. Francis, K.G., Stuckey, P.J.: Explaining circuit propagation. Constraints **19**(1), 1–29 (2014)
10. Hwang, F.K., Richards, D.S., Winter, P.: The Steiner tree problem, Annals of Discrete Mathematics, vol. 53. Elsevier (1992)
11. Kaya, L.G., Hooker, J.N.: A filter for the circuit constraint. In: CP 2006, International Conference on Principles and Practice of Constraint Programming. pp. 706–710. Springer (2006)
12. Laurière, J.: A language and a program for stating and solving combinatorial problems. Artificial Intelligence **10**(1), 29–127 (1978)

13. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard cp modelling language. In: Bessiere, C. (ed.) Principles and Practice of Constraint Programming – CP 2007. pp. 529–543. Springer Berlin Heidelberg (2007)
14. Pesant, G., Gendreau, M., Potvin, J., Rousseau, J.: An exact constraint logic programming algorithm for the traveling salesman problem with time windows. Transportation Science **32**(1), 12–29 (1998)
15. Prosser, P., Unsworth, C.: A connectivity constraint using bridges. In: ECAI 2006: 17th European Conference on Artificial Intelligence. Frontiers in Artificial Intelligence and Applications, vol. 141, pp. 707–708. IOS Press (2006)
16. Prud'homme, C., Fages, J.G., Lorca, X.: Choco Documentation. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S. (2016), `http://www.choco-solver.org`
17. Régin, J.C.: a filtering algorithm for constraints of difference in CSPs. In: AAAI-94, Proceedings of the National Conference on Artificial Intelligence. pp. 362–367. Seattle (1994)
18. Toth, P., Vigo, D.: Vehicle routing: problems, methods, and applications. SIAM (2014)