Iterative aggregation and disaggregation algorithm for pseudo-polynomial network flow models with side constraints

François Clautiaux, Saïdsaïd Hanafi, Rita Macedo, Emilie Voge, Cláudio Alves

Iterative Aggregation and Disaggregation Algorithm for Pseudo-Polynomial Network Flow Models with Side Constraints

François Clautiaux[a,*], Saïd Hanafi[b], Rita Macedo[c], Marie-Émilie Voge[d], Cláudio Alves[e]

[a]*Institut de Mathématiques de Bordeaux (UMR CNRS 5251), Université de Bordeaux, Inria Bordeaux - Sud Ouest*
[b]*LAMIH (UMR CNRS 8201), Université de Valenciennes*
[c]*Laboratoire LEM (UMR CNRS 9221), Université de Lille 3*
[d]*Laboratoire CRISTAL (UMR CNRS 9189), Université Lille 1*
[e]*Departamento de Produção e Sistemas, Escola de Engenharia, Universidade do Minho*

Abstract

This paper develops a general solution framework based on aggregation techniques to solve NP-Hard problems that can be formulated as a circulation model with specific side constraints. The size of the extended Mixed Integer Linear Programming formulation is generally pseudo-polynomial. To efficiently solve exactly these large scale models, we propose a new iterative aggregation and disaggregation algorithm. At each iteration, it projects the original model onto an aggregated one, producing an approximate model. The process iterates to refine the current aggregated model until the optimality is proved.

The computational experiments on two hard optimization problems (a variant of the vehicle routing problem and the cutting-stock problem) show that a generic implementation of the proposed framework allows us to outperform previous known methods.

Keywords:
integer programming, arc-flow integer models, aggregation of integer models, combinatorial optimization

*Corresponding Author. Institut de Mathématiques de Bordeaux, Université de Bordeaux, 351, cours de la Libération - F 33 405 Talence, France

Email address: francois.clautiaux@math.u-bordeaux.fr (François Clautiaux)

1. Introduction

Integer programming models based on network flows with side constraints have been successfully used to solve difficult NP-hard problems by [6, 12, 13, 15]. In such a model, there is a flow that goes through the arcs, visiting some nodes of the network in order to satisfy a given set of constraints. According to the specificities and the objective of each optimization problem, the network that models it may have quite different characteristics. Typically, these problems can be formulated as Mixed Integer Linear Programming (MILP) models, whose variables represent the flow that goes through each of the arcs in the graph. In network flow models designed for solving NP-hard problems, the number of nodes of the network is typically large and depends on the data values of the problem. Therefore, the size of the network is generally not polynomial with regards to the size of the data of the initial problem and neither is the MILP formulation.

To solve a MILP of such large size, an idea is to work on an aggregated network and thus produce an approximate instance of the flow problem. Model aggregation is a classical technique used in several fields of combinatorial optimization (see [17]). One of the most famous examples of aggregation is the state-space relaxation method of [5], and its generalizations (see for example [16]). In integer programming, the most generic aggregation method is the surrogate relaxation, introduced in [11]. In this method, the set of constraints of the original problem is replaced by a single constraint, which is obtained from a linear combination of the relaxed constraints.

Recently, there has been a surge of interest in aggregation techniques in the context of column generation. In [9, 8], the authors aggregate set-partitioning constraints and restrict the sets allowed in the model to those that are *compatible* with the aggregation. The aggregation is then dynamically updated to obtain the optimum of the column generation. In [2], aggregation is applied to a network design model: nodes are aggregated, and an ad-hoc scheme is proposed to disaggregate the current partition, based on min-cut algorithms.

The technique studied in this paper is rather different. It is based on scaling techniques applied to pseudo-polynomial extended MILP formulations. To our knowledge, such aggregation algorithms were applied to network flow models for the first time in [14]. The authors proposed an aggregation algo-

rithm to solve a vehicle routing problem with multiple routes using a time-indexed model. The basic idea is to use a partial discretization and dynamically refine the model by adding new discretization points. This aggregation method was further analyzed by [19]: some theoretical results were proposed, but no clues were given for a practical generic implementation. Similarly to [14] and [19], [3] recently addressed a network design problem using iterative disaggregation techniques in which a scaling algorithm is applied to a pseudo-polynomial formulation. This latter work confirms that iterative scaling is a very effective tool for problems which can be modelled using integer models based on network flows.

In this paper, we propose a general solution framework, the Iterative Aggregation and Disaggregation Algorithm (IADA), which extends the previous method proposed in [14] and [19]. IADA can be applied to large scale arc-flow/circulation models characterized by nodes that correspond to values in a given scale, and arcs that cover elements of a set. An important aspect of our work is its generality. Whereas all aggregation procedures described in previous studies on network-flow problems focus on specific applications, our method applies to a general family of min-cost circulation models which can in turn be applied to a large set of applications. It is based on a given scaling function, which is applied to the arcs' heads and tails. The scaling function determines the level of aggregation as well as the quality of the approximation. It only modifies the possibility of either combining, or not combining, the arcs into feasible paths. As a consequence, a solution that satisfies the side constraints for the original model is obtained directly from the aggregated model. The only question that arises is whether or not the solution respects the flow-conservation constraints. When this is not the case, the method iteratively disaggregates the network until convergence is proved.

Our computational experiments on two hard combinatorial optimization problems (a variant of the vehicle routing problem and the cutting-stock problem) show that a generic implementation of our framework allows us to outperform previously studied methods.

The paper is organized as follows. In Section 2, we define our minimum cost circulation model, and in Section 3 we provide theoretical results related to our aggregation scheme. Section 4 describes the general framework for iterative aggregation and disaggregation of flow/circulation models. In Section 5, we describe two practical problems that can be solved with this algorithm, and present our computational results. Finally, some conclusions are drawn in section 6.

2. Minimum cost circulation integer models

In this section, we describe the class of integer linear models studied in this paper, and describe precisely two instances of such formulations. Throughout the paper, we use the following notation. Let $\mathcal{I}$ be a set of elements (representing customers, items, jobs, ..., *etc.*) and $\mathcal{V} = \{0, \ldots, W\}$ be the ordered set of nodes representing values in a given scale (time, position, ..., *etc.*). Let $G = (\mathcal{V}, \mathcal{A}, s, t, e, c)$ be a directed acyclic multigraph, where $\mathcal{A}$ is the set of arcs, $s : \mathcal{A} \to \mathcal{V}$ assigns to each arc its tail, $t : \mathcal{A} \to \mathcal{V}$ assigns to each arc its head, $e : \mathcal{A} \to \mathcal{P}(\mathcal{I})$ associates each arc a to a subset $e(a) \subseteq \mathcal{I}$, and $c : \mathcal{A} \to \mathbb{R}$ associates a to its cost. Set $e(a)$ represents the elements of $\mathcal{I}$ *covered* by arc a. We assume that $s(a) \leq t(a)$, $\forall a \in \mathcal{A}$, except for a special arc a^* of tail W and head 0. The element $i^* \in \mathcal{I}$ covered by a^* is a dummy element that we will use in our model. To simplify the notation, we may refer to an arc a by the triple $(s(a), t(a), e(a))$. The source of the network is node 0, and its sink is node W. Set $\mathcal{A}$ contains also arcs $(v, v', \emptyset)$, for all $v \in \mathcal{V} \setminus \{W\}$, $v' \in \mathcal{V} \setminus \{0\}$, where v and v' are any two consecutive nodes of $\mathcal{V}$. These arcs do not cover any element of $\mathcal{I}$ and are called *dummy arcs*. These arcs have a cost equal to zero, and thus can be used to complete paths when regular arcs are not sufficient to build a path from 0 to W. For each $i \in \mathcal{I}$, $\mathcal{A}(i) \subseteq \mathcal{A}$ denotes the subset of arcs covering element i (i.e., $\mathcal{A}(i) = \{a \in \mathcal{A} : i \in e(a)\}$). A circulation x in graph G *covers* an element $i \in \mathcal{I}$ if $x_a > 0$ for some $a \in \mathcal{A}(i)$.

We focus on a specific class of integer models that can be described as a minimum-cost circulation problem with linking bound constraints. The model takes as an input a multigraph G, and lower and upper bounds $\ell_i, u_i \in \overline{\mathbb{R}} \equiv \mathbb{R} \cup \{-\infty, +\infty\}$ for all elements $i \in \mathcal{I}$. We use variables $x_a, \forall a \in \mathcal{A}$, which correspond to the flow going through the arcs of $\mathcal{A}$. The min-cost circulation model can thus be written as follows.

$$\min \quad \sum_{a \in \mathcal{A}} c(a) x_a \tag{1}$$

$$\text{s.t.} \quad \ell_i \leq \sum_{a \in \mathcal{A}(i)} x_a \leq u_i, \qquad \forall i \in \mathcal{I}, \tag{2}$$

$$\sum_{a' \in \mathcal{A} : s(a') = v} x_{a'} = \sum_{a \in \mathcal{A} : t(a) = v} x_a, \qquad \forall v = 0, \ldots, W \tag{3}$$

$$x_a \in \mathbb{N}, \qquad \forall a \in \mathcal{A} \tag{4}$$

Note that the flow going through arc a^* represents the value of the circulation. This value can be fixed by setting $\ell_{i^*} = u_{i^*}$.

In the remainder of the paper, for a multigraph G, an element set $\mathcal{I}$, and two bound vectors $\ell, u \in \overline{\mathbb{R}}^{|\mathcal{I}|}$, we will denote the model (1)–(4) applied to this data as $X(G, \ell, u, \mathcal{I})$, and the optimal value of model $X(G, \ell, u, \mathcal{I})$ as $\text{val}(G, \ell, u, \mathcal{I})$.

In order to show the generality of this model and illustrate the notations above, we describe two examples that represent two very different applications that can be modelled with (1)–(4): a routing problem and a packing problem.

2.1. The vehicle routing problem with multiple trips

Our first example, taken from [14], is a routing problem where K vehicles perform several small routes during the same planning period (workday). In this problem, the goal is to lexicographically maximize the number of customers visited and minimize the total distance travelled. The deliveries must be performed during a workday lasting W units of time. The customers can only be served during a given time window.

Let $\mathcal{I}$ be the set of customers to visit. Each possible route r is characterized by a sequence $\mathcal{J} \subseteq \mathcal{I}$ of customers, a duration $\text{length}(r)$ and a global time-window $[d_r^-, d_r^+]$ deduced from the individual time-windows of the served customers. The network G is constructed as follows: the node set $\mathcal{V}$ corresponds to time instants $\{0, \ldots, W\}$ of the workday, and the arcs of $\mathcal{A}$ correspond to the different possible (precomputed) routes. Each route r is modeled in the network as a list of arcs $a_d = (d, d + \text{length}(r), \mathcal{J})$, $d \in [d_r^-, d_r^+]$, representing the selection of the route that goes through $\mathcal{J}$, beginning at time d and finishing at time $d + \text{length}(r)$. Therefore, a path supporting a unit flow represents the workday of a vehicle. The cost $c(a)$ of an arc is computed as follows: let r_a be the route related to arc a, $\text{length}(r_a)$ its length, and $n(r_a)$ the number of customers serviced by route r_a. The cost of arc a is $c(a) = \text{length}(r_a) - M * n(r_a)$, where M is a large constant integer value. For each customer $i \in \mathcal{I}$, we set the bounds as follows: $\ell_i = 0$ and $u_i = 1$. To take into account the constraint on the number of vehicles, we set $u_{a^*} = K$ and since using a vehicle has no additional cost, we set $c(a^*) = 0$.

2.2. The cutting-stock problem

We also consider the cutting-stock problem, where an integer flow of minimum value has to cover the whole set of elements. The corresponding

model, with a different formalism, is used in [6]. In this problem, there is a set $\mathcal{I}$ of items i, each of size $w_i \in \mathbb{N}$ and demand $\ell_i \in \mathbb{R}$, and a unique size $W \in \mathbb{N}$ of a bin. The goal is to find the minimum number z of bins needed to pack ℓ_i times each item i of $\mathcal{I}$.

The multigraph is built as follows. There is one node v for each position $\{0, 1, \ldots, W\}$ in the bin. For each item i, the corresponding arc set $\mathcal{A}(i)$ is built as follows: for $j = 0, \ldots, W - w_i$, add an arc $(j, j + w_i, i)$ that covers only i. For each element $i \in \mathcal{I}$, the bounds are computed as follows. The values ℓ_i are those defined in the original cutting-stock problem, and $u_i = +\infty$, $\forall i \in \mathcal{I}$. The cost of each arc $a \in \mathcal{A}$ is 0, except for arc a^*, whose cost is 1. Since the number of bins is not limited, we set $\ell_{a^*} = 0$ and $u_{a^*} = +\infty$. The objective is to minimize the value of the circulation. Practically speaking, many arcs can be removed by preprocessing using dominance rules. These rules are described precisely in [6].

3. Aggregation of network flow models

In this section, we formally introduce the concept of aggregation applied to our network circulation model. We consider two types of aggregation: heuristic and conservative. In the first case, some valid paths are not valid anymore, and solving the model obtained leads to a heuristic solution. In the second case, some paths are added to the model, and thus a relaxation is obtained.

We generalize the results of [19], which made two distinct cases of profit maximization and cardinality cover. As explained above, these two cases can both be expressed as a minimum-cost flow problem with linking bounds.

3.1. Aggregation

We now give the basic definitions that are used throughout the paper. The three following definitions are modified versions of those from [19]. More precisely, we define more formally the notion of valid aggregation. We first define the notion of *discretization function*, which allows the projection of a discrete set onto another one. We only consider non-decreasing functions, since they may otherwise break the structure of the model. Without any loss of generality, we will assume that the target set $\mathcal{S}$ of the discretization is a subset of the initial set $\mathcal{V}$.

Definition 1 (discretization function)**.** *For a given discrete node set* $\mathcal{V} = \{0, \ldots, W\}$*, and a target set of discrete values* $\mathcal{S} \subseteq \mathcal{V}$ *such that* $\{0, W\} \subseteq \mathcal{S}$*, a discretization function* ρ *is a non-decreasing function defined from* $\mathcal{V}$ *to* $\mathcal{S}$*.*

An *aggregation* of a multigraph is obtained by applying two discretization functions to the two extremities of each of its arcs.

Definition 2 (aggregated network)**.** *Let* $G = (\mathcal{V}, \mathcal{A}, s, t, e, c)$ *be a network flow multigraph and* (ρ_1, ρ_2) *be a pair of discretization functions defined from* $\mathcal{V}$ *to* $\mathcal{S}$*. The aggregated network associated to the aggregation function* $\Psi = (\rho_1, \rho_2)$ *is computed as follows:* $\Psi(G) = (\mathcal{S}, \mathcal{A}, \rho_1 \circ s, \rho_2 \circ t, e, c)$*. Aggregation* Ψ *is valid if for any pair of consecutive values* v *and* v' *in* $\mathcal{S}$*, there exists a dummy arc* $a \in \mathcal{A}$ *such that* $\rho_1 \circ s(a) = v$ *and* $\rho_2 \circ t(a) = v'$*.*

Note that the aggregated network may contain loops, *i.e.* when $\rho_1 \circ s(a) = \rho_2 \circ t(a)$. This is generally bad for the method. However, depending on the side constraints, useful information can still be obtained from the aggregated network when loops are created.

The basic idea of this paper is to work on an aggregated model $X(\Psi(G), \ell, u, \mathcal{I})$, which is hopefully easier to optimize, since it has fewer constraints and variables. Theoretically, such an aggregation only reduces the number of nodes (and thus the number of constraints in the corresponding min-cost circulation models), and not the number of arcs (variables). Practically speaking, after aggregation many arcs covering the same set of elements will have the same tail and head. Therefore, only one replication will be kept, reducing the number of arcs/variables. For the sake of simplicity, our theoretical characterization of aggregation will not consider this arc reduction phase, which does not modify the optimal value of the model. In our framework, aggregation is used to compute iteratively primal bounds, and dual bounds that converge towards the optimum. Therefore, we will only consider aggregations that lead to an over-constrained model, or to a relaxation.

Definition 3 (conservative and heuristic aggregations)**.** *Let* $\Psi = (\rho_1, \rho_2)$ *be an aggregation function, where* ρ_1 *and* ρ_2 *are defined from* $\mathcal{V}$ *to a given set* $\mathcal{S} \subseteq \mathcal{V}$*. We say that* Ψ *is a* conservative aggregation *if for any pair of arcs* a_1 *and* a_2*,* $t(a_1) \leq s(a_2)$ *implies that* $\rho_2 \circ t(a_1) \leq \rho_1 \circ s(a_2)$*. Similarly,* Ψ *is a* heuristic aggregation *if for any pair of arcs* a_1 *and* a_2*,* $\rho_2 \circ t(a_1) \leq \rho_1 \circ s(a_2)$ *implies that* $t(a_1) \leq s(a_2)$*.*

Without loss of generality, in the remainder of this paper, we will consider only discretization functions ρ such that $\rho(0) = 0$ and $\rho(W) = W$.

Obviously, if one solves exactly the model obtained after a heuristic aggregation, a feasible solution is found. On the other hand, any dual bound for the problem obtained using the conservative aggregation is also a dual bound for the initial model. To illustrate the notions of conservative and heuristic aggregations, consider the two following discretization functions, ρ^- and ρ^+, defined from $\mathcal{V}$ to a given target set $\mathcal{S}$, where $\rho^-(v) = \max\{p \in \mathcal{S} : p \leq v\}$ and $\rho^+(v) = \min\{p \in \mathcal{S} : p \geq v\}$, for all $v \in \mathcal{V}$. Function $\Psi_h = (\rho^-, \rho^+)$ is a heuristic aggregation and function $\Psi_c = (\rho^-, \rho^-)$ is a conservative aggregation. In the following, $\mathrm{Id} = (\rho^=, \rho^=)$ represents the identity aggregation function, where $\rho^=(v) = v$. By definition, the identity aggregation is both heuristic and conservative.

We now give a new and more precise characterization of the combinations of discretization functions that produce a conservative aggregation.

Lemma 1. *Let ρ_1 and ρ_2 be two valid discretization functions defined from set $\mathcal{V}$ to set $\mathcal{S} \subseteq \mathcal{V}$. A valid aggregation $\Psi = (\rho_1, \rho_2)$ is conservative if and only if $\rho_2(v) = \rho_1(v), \forall v \in \mathcal{V}$, and ρ_1 is surjective.*

Proof. Necessary condition: consider the path $\mu = a_1, \ldots, a_k$ composed of all dummy arcs. If $\exists v \in \mathcal{V}$ such that $\rho_2(v) \neq \rho_1(v)$ then $\exists j \in \{1, \ldots, k\}$ such that $t(a_j) = s(a_{j+1})$ and $\rho_2 \circ t(a_j) \neq \rho_1 \circ s(a_{j+1})$. If $\rho_2 \circ t(a_j) > \rho_1 \circ s(a_{j+1})$, Ψ is clearly not conservative. If $\rho_2 \circ t(a_j) < \rho_1 \circ s(a_{j+1})$, we can conclude that Ψ is not valid because there cannot be a dummy arc between $\rho_2(v)$ and its successor in $\mathcal{S}$. Suppose such an arc a_p exists. This arc a_p satisfies $\rho_1 \circ s(a_p) = \rho_2 \circ t(a_j) < \rho_1 \circ s(a_{j+1})$, which implies that $p < j + 1$ (because ρ_1 is non-decreasing). This arc a_p also satisfies $\rho_2 \circ t(a_p) > \rho_2 \circ t(a_j)$, which implies that $p > j$ (because ρ_2 is non-decreasing). Therefore, there cannot be such an arc, and Ψ cannot be valid. If ρ_1 is not surjective, then there are nodes $v \in \mathcal{S}$ such that there is no dummy arc going in or out of v.

Sufficient condition: for any pair of arcs a_1 and a_2 such that $t(a_1) \leq s(a_2)$, since ρ_1 is increasing, we obtain $\rho_2 \circ t(a_1) = \rho_1 \circ t(a_1) \leq \rho_1 \circ s(a_2)$, which is the conservative aggregation condition. The fact that ρ_1 is surjective and increasing directly implies that for any consecutive pair of nodes $v, v' \in \mathcal{S}$, there is an arc a such that $\rho_1 \circ s(a) = v$ and $\rho_1 \circ t(a) = v'$. □

3.2. Approximations and bounds

We start this section by defining the notions of *conflict graph* and *conflict-difference graph*, initially proposed in [19]. They use the notion of *conflict* between two arcs. Two arcs are said to be in conflict if they cannot be used in the same path. This allows us to give a bound for the ratio between the optimal solution values of the non-aggregated and aggregated models. We need them in the next section to design strategies for computing a suitable initial aggregation.

Definition 4 (conflict between two arcs)**.** *Two arcs a_1 and a_2 are* in conflict *in graph G if $[s(a_1), t(a_1)] \cap [s(a_2), t(a_2)] \neq \emptyset$.*

Definition 5 (conflict graph)**.** *Let $G = (\mathcal{V}, \mathcal{A}, s, t, e, c)$ be a multigraph. The* conflict graph *associated to graph G is the interval graph $\widehat{G} = (\mathcal{A}, \mathcal{C})$ such that the edge $\{a_1, a_2\}$ belongs to $\mathcal{C}$ iff arcs $a_1, a_2 \in A$ are in conflict in graph G.*

Definition 6 (conflict difference graph)**.** *Let $G = (\mathcal{V}, \mathcal{A}, s, t, e, c)$ be a network flow multigraph and Ψ^+ and Ψ^- be two aggregation functions such that $\mathcal{C}^-$, the arc set of $\widehat{\Psi^-(G)}$, is a subset of $\mathcal{C}^+$, the arc set of $\widehat{\Psi^+(G)}$. The* conflict difference graph *of graph G related to Ψ^+ and Ψ^- is the graph $\widehat{G}(\Psi^+, \Psi^-) = (\mathcal{A}, \mathcal{C}^+ \setminus \mathcal{C}^-)$.*

The conflict difference graph represents the conflicts present in $\Psi^+(G)$ but not in $\Psi^-(G)$. If Ψ^+ is an heuristic aggregation, the set of conflicts of $\Psi^+(G)$ contains the set of conflicts of graph G: the conflict difference graph is thus $\widehat{G}(\Psi^+, \mathrm{Id})$. Similarly, a conservative aggregation removes conflicts without creating any. If Ψ^- is a conservative aggregation, the conflict difference graph is $\widehat{G}(\mathrm{Id}, \Psi^-)$.

Let us consider two examples of cardinality cover minimization problems, whose original multigraphs are illustrated in Figures 1.*a*) and 2.*a*), and the conservative and heuristic aggregation functions $\Psi_c = (\rho^-, \rho^-)$ and $\Psi_h = (\rho^-, \rho^+)$, respectively. Figures 1.*b*) and 2.*b*) represent the aggregated models when Ψ_c and Ψ_h are applied, and Figures 1.*c*), 2.*c*) and 1.*d*), 2.*d*) represent the corresponding conflict graphs. The conflict difference graphs of G and G', related to the original and aggregated multigraphs, are represented in Figures 1.*e*) and 2.*e*) respectively. In graph G, the minimum number of paths that cover all arcs is equal to 3, while in $\Psi_c(G)$ it is equal to 2. This happens as a number of conflicts was removed (Figure 1.*e*). For example, a

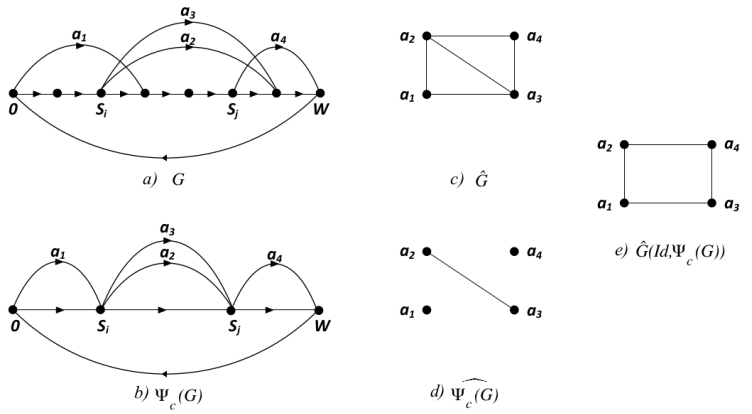

Figure 1: Conservative aggregation. Subfigure a) is the original multigraph G, with its conflict graph c). Subfigure b) is the aggregated multigraph obtained using $\mathcal{S} = \{0, \mathcal{S}_i, \mathcal{S}_j, W\}$, with the corresponding conflict graph d). The conflict difference graph is e).

path composed of arcs a_1, a_2 and a_3 is feasible in the aggregated model but not in the original one. On the other hand, in G', the minimum number of paths that cover all arcs is equal to 2, while in $\Psi_h(G')$ it is equal to 3. This time, a number of conflicts was added (Figure 2.e). For example, a path composed of arcs a_1, a_3 and a_5 is feasible in the original model, but not in the aggregated one.

The cliques in conflict-difference graphs play an important role in the remainder of this section. This role is described in the following lemma, which is a rewriting of a lemma by [19].

Lemma 2. *For two given aggregation functions Ψ^+ and Ψ^- and a network flow multigraph G, let $\omega(\Psi^+, \Psi^-)$ be the size of the maximum clique in $\widehat{G}(\Psi^+, \Psi^-)$. Let $\mathcal{A}^\mu$ be the arc set supporting a $0W$–path μ in $\Psi^-(G)$. At most $\omega(\Psi^+, \Psi^-)$ $0W$–paths are necessary in $\Psi^+(G)$ to cover the arcs of $\mathcal{A}^\mu$.*

Lemma 2 allows us to determine the approximation factors and worst-case performances obtained by using respectively a heuristic or a conservative aggregation in some specific cases. In the general case (*i.e.* random values of $c(a)$), no approximation ratio can be proved, since the optimal value may be zero and the problem is NP-hard. For some combinations of constraints, feasibility issues can occur: there may be a feasible solution after a conservative

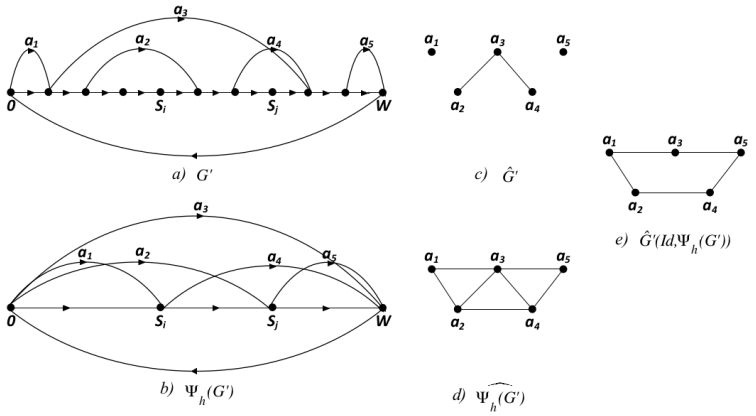

Figure 2: Heuristic aggregation. Subfigure *a*) is the original multigraph G, with its conflict graph *c*). Subfigure *b*) is the aggregated multigraph obtained using $\mathcal{S} = \{0, \mathcal{S}_i, \mathcal{S}_j, W\}$, with the corresponding conflict graph *d*). The conflict difference graph is *e*).

aggregation, while no such solution exists for the initial problem. The opposite can occur with a heuristic aggregation. However, in specific cases where maintaining feasibility is not an issue, it is possible to prove that the ratio between $\text{val}(\Psi(G), \ell, u, \mathcal{I})$ and $\text{val}(G, \ell, u, \mathcal{I})$ is bounded by $\omega(\Psi, \text{Id})$. Since the clique size is not bounded by any constant in general, the approximation factor is not a constant value.

Proposition 1. *Let $G = (\mathcal{V}, \mathcal{A}, s, t, e, c)$ be a network, $\mathcal{I}$ a finite set of elements, and $\ell \in \overline{\mathbb{R}}^{|\mathcal{I}|}$, $u \in \overline{\mathbb{R}}^{|\mathcal{I}|}$ two vectors. Let also Ψ_h be a heuristic aggregation and $\omega(\Psi_h, \text{Id})$ be the size of the maximum clique in $\widehat{G}(\Psi_h, \text{Id})$. If the two following conditions hold:*

1. *$c(a) \geq 0, \forall a \in \mathcal{A}$, or $c(a) \leq 0$, $\forall a \in \mathcal{A}$*
2. *$\ell_i = 0$, $\forall i \in \mathcal{I}$ or $u_i = +\infty$, $\forall i \in \mathcal{I}$*

then $val(\Psi_h(G), \ell, u, \mathcal{I}) - val(G, \ell, u, \mathcal{I}) \leq (\omega(\Psi_h, \text{Id}) - 1) * |val(G, \ell, u, \mathcal{I})|$.

Proof. First, note that under the first assumption, if $\text{val}(G, \ell, u, \mathcal{I}) = 0$, then the solution is a circulation of value zero (or using dummy arcs only). In this case, any heuristic aggregation will lead to this solution, and the proposition is true. In the following, we can assume without loss of generality that $\text{val}(G, \ell, u, \mathcal{I}) \neq 0$,

Consider an optimal circulation for $X(G, \ell, u, \mathcal{I})$ such that the flow through a^* is z^*. If this optimal circulation is feasible in $X(\Psi_h(G), \ell, u, \mathcal{I})$, then the

proposition directly follows. If it is not the case, we show how we can construct a feasible solution. We know that this circulation can be decomposed into z^* circuits. Using Lemma 2, we know that the elements of each of these circuits can be covered by less than $\omega(\Psi_h, \mathrm{Id})$ circuits in $\Psi_h(G)$. The way a feasible solution for $X(\Psi_h(G), \ell, u, \mathcal{I})$ is obtained depends on which part of condition 2 is true.

If $u_{i^*} = +\infty$, then the solution obtained by using $\omega(\Psi_h, \mathrm{Id})$ circuits is directly feasible because it is always possible to increase the value of the circulation, and the other bound constraints are not modified. Without loss of generality, we assume that we are in the case $c(a) \geq 0, \forall a \in \mathcal{A}$, since the problem is otherwise unbounded. The additional cost of the heuristic solution is $(\omega(\Psi_h, \mathrm{Id}) - 1)c(a^*)$ (the elements covered are the same). The worst case occurs when $c(a) = 0, \forall a \in \mathcal{A} \setminus \{a^*\}$. In this case, the difference between the optimal value and this specific heuristic value is $(X(\Psi_h(G), \ell, u, \mathcal{I}) - 1)\mathrm{val}(G, \ell, u, \mathcal{I})$. Since the optimal solution of $X(\Psi_h(G), \ell, u, \mathcal{I})$ is better than this specific heuristic, one obtains $\mathrm{val}(\Psi_h(G), \ell, u, \mathcal{I}) - \mathrm{val}(G, \ell, u, \mathcal{I}) \leq (\omega(\Psi_h, \mathrm{Id}) - 1) * \mathrm{val}(G, \ell, u, \mathcal{I})$.

If $\ell_i = 0, \forall i \in \mathcal{I}$, then we can drop some arcs to obtain a feasible solution. Without loss of generality, we assume that $c(a^*) = 0$ (since it is always possible to use dummy arcs to increase the value of the circulation). Because it may happen that $u_{i^*} \in \mathbb{R}$, it may not be possible to use all these circuits. In the worst case, $z^* = u_{i^*}$. In this case, one can recover a valid solution by dropping at most $(\omega(\Psi_h, \mathrm{Id}) - 1)z^*$ circuits. This is always possible under the assumption $\ell(i) = 0, \forall i \in \mathcal{I}$. The worst case occurs when all circuits have the same value. In this case, one obtains $\mathrm{val}(\Psi_h(G), \ell, u, \mathcal{I}) - \mathrm{val}(G, \ell, u, \mathcal{I}) \leq (\omega(\Psi_h, \mathrm{Id}) - 1) * |\mathrm{val}(G, \ell, u, \mathcal{I})|$.

□

Similar results stand for the lower bounds. There is an additional condition to check in this case: the solution value must be different from zero. Under the other conditions, this solution corresponds to a flow of value zero, or using dummy arcs only. In this case, the problem is trivial, thus the condition does not weaken the result.

Proposition 2. *Let $G = (\mathcal{V}, \mathcal{A}, s, t, e, c)$ be a network, $\mathcal{I}$ a finite set of elements, $\ell \in \overline{\mathbb{R}}^{|\mathcal{I}|}$ and $u \in \overline{\mathbb{R}}^{|\mathcal{I}|}$ two vectors. Let also Ψ_c be a conservative aggregation and $\omega(\mathrm{Id}, \Psi_c)$ be the size of the maximum clique in $\widehat{G}(\mathrm{Id}, \Psi_c)$. If the three following conditions hold:*

1. $c(a) \geq 0, \forall a \in \mathcal{A}$, *or* $c(a) \leq 0$, $\forall a \in \mathcal{A}$
2. $\ell_i = 0$, $\forall i \in \mathcal{I}$ *or* $u_i = +\infty$, $\forall i \in \mathcal{I}$
3. $val(G, \ell, u, \mathcal{I}) \neq 0$

then $val(G, \ell, u, \mathcal{I}) - val(\Psi_c(G), \ell, u, \mathcal{I}) \leq (\omega(\mathrm{Id}, \Psi_c) - 1) * |val(G, \ell, u, \mathcal{I})|$.

Proof. The proof is similar to the proof of Proposition 1. Network G is now the multigraph with the higher number of conflicts and $\Psi_c(G)$ the one with the lower number of conflicts since Ψ_c is conservative. The only difference is that the property does not hold if $\mathrm{val}(G, \ell, u, \mathcal{I}) = 0$, because we cannot guarantee that the relaxation will not result in a negative cost.

□

We have shown that cliques in conflict difference graphs play a major role in the theoretical quality of the aggregation. We now state the accurate correspondence between cliques in the conflict difference graph $\widehat{G}(\Psi_h, \mathrm{Id})$ and conflicts in the initial graph. For a given arc set $\mathcal{A}$, a scale $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$ and a value $j < |\mathcal{S}| - 1$, let $\mathcal{A}^j = \{a \in \mathcal{A} : S_j \leq s(a) \leq S_{j+1}$ or $\mathcal{S}_j \leq t(a) \leq \mathcal{S}_{j+1}\}$, *i.e.*, $\mathcal{A}^j$ is the set of arcs with at least one extremity between $\mathcal{S}_j$ and $\mathcal{S}_{j+1}$. We use the classical notation $\widehat{G}[\mathcal{A}^j]$ for the subgraph induced by the vertex set $\mathcal{A}^j$ of $\widehat{G}$.

Lemma 3. *Let* $G = (\mathcal{V}, \mathcal{A}, s, t, e, c)$ *be a network flow multigraph,* $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_k\} \subseteq \mathcal{V}$ *and* $\Psi_h = (\rho^-, \rho^+)$ *be a heuristic aggregation. Let* η^j *be the set of stable sets in the conflict subgraph* $\widehat{G}[\mathcal{A}^j]$*, and* $\chi(\Psi_h, \mathrm{Id})$ *be the set of cliques in* $\widehat{G}(\Psi_h, \mathrm{Id})$*. There is a bijection between the set* $\chi(\Psi_h, \mathrm{Id})$ *and the set* $\bigcup_{j=1,\ldots,k-1} \eta^j$.

Proof. We first show that for each element of $\chi(\Psi_h, \mathrm{Id})$, there is a unique element of the set $\bigcup_{j=1,\ldots,k-1} \eta^j$. Each clique $Q \in \chi(\Psi_h, \mathrm{Id})$ is clearly related to a unique stable set $\{a_1, \ldots, a_p\}$ of $\widehat{G}$ (otherwise there cannot be a clique in the conflict difference graph). Without loss of generality, we assume that the arcs are sorted by increasing values of tail. It remains to be shown that the stable set $\{a_1, \ldots, a_p\}$ associated to Q is such that $\mathcal{S}_j \leq t(a_1) \leq s(a_p) \leq \mathcal{S}_{j+1}$ for a given j. Let us assume that this is not the case. This means that $t(a_1) < \mathcal{S}_j$ or $s(a_p) > \mathcal{S}_{j+1}$. In both cases, $\rho^+ \circ t(a_1) \leq \rho^- \circ s(a_p)$ and therefore a_1 and a_p cannot be in conflict in the aggregated multigraph, which contradicts the initial assumption. Therefore, all arcs of the stable set belong to $\widehat{G}[\mathcal{A}^j]$. Thus

we have shown that each clique of $\chi(\Psi_h, \mathrm{Id})$ is related to a unique stable set of η^j for a given j.

Now let $\pi_j = \{a_1, \ldots, a_p\}$ for a given j be a stable set of η^j. When the aggregation Ψ_h is performed, the arcs a_i such that $\mathcal{S}_j \leq s(a_i) \leq t(a_i) \leq \mathcal{S}_{j+1}$ now connect $\mathcal{S}_j = \rho^- \circ s(a_i)$ and $\mathcal{S}_{j+1} = \rho^+ \circ t(a_i)$ and thus their associated nodes form a clique in $\widehat{G}(\mathrm{Id}, \Psi_h)$. The possible unique arc a_i of the set such that $s(a_i) < \mathcal{S}_j \leq t(a_i) \leq \mathcal{S}_{j+1}$ now connects the nodes $\mathcal{S}_{j'}$ and $\mathcal{S}_{j+1}$ ($j' < j$). Similarly, the possible unique arc a_i of the set such that $\mathcal{S}_j \leq s(a_i) \leq \mathcal{S}_{j+1} < t(a_i)$ now connects $\mathcal{S}_j$ and a node $\mathcal{S}_{j'}$ such that $j' > j$. Consequently these two possible arcs are in conflict with all other arcs of π_j in the aggregated multigraph. Therefore, all arcs of this set form a clique in $\widehat{G}(\mathrm{Id}, \Psi_h)$.

□

Practically speaking, since the conflict graph is an interval graph, the maximum stable set in each graph $\widehat{G}[\mathcal{A}^j]$ can be computed in linear time. This lemma led us to design a method for computing an initial scale in our aggregation framework, together with the following proposition.

Proposition 3. *Let G be a network multigraph, $\Psi = (\rho_1, \rho_2)$ be a conservative aggregation, and $X(G, \ell, u, \mathcal{I})$ be a min-cost circulation model. A solution for the aggregated model $X(\Psi(G), \ell, u, \mathcal{I})$ may only be infeasible for $X(G, \ell, u, \mathcal{I})$ if there are two arcs a and a' in the solution such that $\rho_1 \circ t(a) = \rho_1 \circ s(a')$.*

Proof. Since ρ_1 is non-decreasing, $\rho_1 \circ t(a) < \rho_1 \circ s(a')$ implies $t(a) < s(a')$. Therefore, a path can use both a and a' in $X(G, \ell, u, \mathcal{I})$. If $\rho_1 \circ t(a) > \rho_1 \circ s(a')$ then since the aggregation is conservative, no path can use both a and a' in $X(G, \ell, u, \mathcal{I})$. Therefore, only the case $\rho_1 \circ t(a) = \rho_1 \circ s(a')$ remains and can lead to an infeasibility. □

4. A general framework for iterative aggregation and disaggregation of network flow models

The proposed aggregation method is not static. Once a first aggregation has been performed, it is iteratively refined until an optimal solution is found. We now propose our general algorithmic framework, the Iterative Aggregation and Disaggregation Algorithm (IADA), which may be applied to any problem that can be formulated as our circulation model.

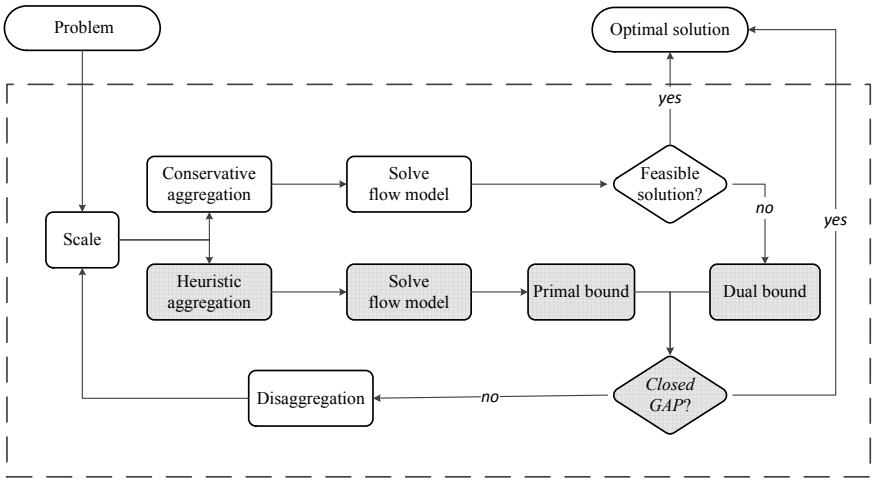

Figure 3: Overview of the method

We designed two versions of our algorithm, depending on whether or not a heuristic aggregation is used. Figure 3 presents our general framework, where steps in grey are optional. The algorithm can be summarized as follows: given the original network flow model for the considered problem, an aggregated model is created. At each iteration, two aggregated flow models are solved, one corresponding to a relaxation (conservative aggregation) and the other to a restriction (heuristic aggregation) of the original problem. The conservative and heuristic aggregated models respectively provide a dual and a primal bound. At each iteration, the current model is updated by disaggregation of a subset of its nodes, in order to exclude the current solution in the subsequent iterations. The process iterates until either a feasible solution of the relaxed model is obtained or the gap between the dual and primal bounds is closed. Algorithm 1 summarizes our method. There are three main steps to consider in the algorithm: finding an initial scale of aggregated nodes, checking the feasibility of a solution and defining a disaggregation scheme.

4.1. Computing a suitable initial scale

On the one hand, considering an initial scale $\mathcal{S}$ with fewer elements implies having a smaller model, with fewer variables (arcs) and constraints (nodes). On the other hand, a coarser scale implies that the quality of the relaxation/heuristic obtained is weaker. Several methods can be used to determine this initial scale. Since the size of the model obtained is strongly

correlated with the cardinality of $\mathcal{S}$, each of them is parameterized by a given integer k. We propose the following three generic methods to compute the initial scale.

The first generic method is called **Regular Initial Scale (RIS)**. It does not rely on any information on the data. It consists of selecting the following set of values: $\mathcal{S} = \{0, k, 2k, \ldots, W - (W \mod k), W\}$. This aggregation is efficient when the data are uniformly distributed on the initial scale.

The second generic method is called **Max Extremities (ME)**. It is inspired from Proposition 3. The idea is to keep the nodes with the largest number of incident arcs. They are computed as follows: the nodes in $\mathcal{V}$ are sorted by the decreasing number of incident arcs. The k first nodes of the list and nodes 0 and W are then selected. This aggregation is efficient when there are several vertices with a large number of in-going and out-going arcs.

The last generic method is called **Max Crossing (MC)**. It is inspired from Proposition 1 and Lemma 3. The idea is to keep the nodes v for which $s(a) \leq v \leq t(a)$ for as much arcs a as possible. The target node set is computed as follows. Nodes $v \in \mathcal{V}$ are sorted by decreasing number of arcs a such that $s(a) \leq v \leq t(a)$. The k first nodes of the list and nodes 0 and W are then selected. This aggregation avoids having a large number of arc tails or heads aggregated in the same node.

4.2. Feasibility checking

It is not straightforward to determine if a solution of the current aggregated model is feasible or not for the original one, since a solution of a network flow integer model corresponds to a set of arcs. We will now show that it is not straightforward to transform this set of arcs in the aggregated model into a set of valid paths in the original model.

We will now define precisely the *feasibility checking problem*. The solution of a network flow model is given by the values of flow variables on the arcs: $\{x_a : a \in \mathcal{A}\}$, and the flow through arc a^*, that we denote by z^*. To ease the presentation, we introduce the following notation: for a given arc a, and a given aggregation $\Psi_c = (\rho^1, \rho^2)$, we note $\text{eq}(a) = \{a' \in \mathcal{A}(e(a)) : \rho^1 \circ s(a) = \rho^1 \circ s(a'), \rho^2 \circ t(a) = \rho^2 \circ t(a')\}$, *i.e.* the set of arcs that are equivalent to a with respect to Ψ_c.

Problem 1 (Feasibility problem of an aggregated solution)**.** *Given a multigraph G, an aggregated multigraph $\Psi_c(G)$, and a circulation $\{x_a : a \in \mathcal{A}\}$ in $\Psi_c(G)$, the problem of feasibility checking involves determining if there exists a feasible circulation $\{\hat{x}_a : a \in \mathcal{A}\}$ for G, such that $\forall a \in \mathcal{A}, \sum_{a' \in eq(a)} \hat{x}_{a'} = x_a$.*

To show that this problem is NP-hard, we reformulate it into a high-multiplicity scheduling problem.

Problem 2 ($P|r_j|\sum U_j$)**.** *Let $\mathcal{J}$ be a set of activities j, each having a processing time p_j, a release date r_j, a deadline d_j and a number of repetition m_j. We assume that all data are integer. The problem is to schedule the jobs of $\mathcal{J}$ on M machines in such a way that each activity j begins after r_j. If a job j is late then $U_j = 1$. The objective is to minimize the number of late jobs (*i.e. *the number of jobs j ending after d_j).*

This problem is strongly NP-hard, since it generalizes $P||C_{\max}$, which is itself strongly NP-hard ([10]). From the feasibility problem, we construct the scheduling problem as follows: let $\Psi(G) = (\mathcal{S}, \mathcal{A}, \rho_1 \circ s, \rho_2 \circ t, e, c)$ be the current aggregated network. For each arc $a \in \mathcal{A} \setminus \{a^*\}$ such that $x_a > 0$, we create an activity a with a processing time $p_a = t(a) - s(a)$, a release date $r_a = \min\{s(a') : a' \in \text{eq}(a)\}$ and a due date $d_a = \max\{t(a') : a' \in \text{eq}(a)\} - p_a$. The repetition m_a is set to x_a. The number of machines $m = |M|$ is set to z^*, the value of x_{a^*}. The answer to the feasibility problem is yes if there is a solution such that $\sum_{j \in \mathcal{J}} U_j = 0$, and otherwise the answer is no.

4.2.1. A MIP formulation for the unit case

We propose a MIP formulation for the feasibility version of the $P|r_j|\sum U_j$ problem for the case where the multiplicity of each arc is one. In this case, z^* is bounded by the number of arcs used, and an arc supports, at most, one unit of flow. Let $\mathcal{A}^* = \{a \in \mathcal{A} : x_a > 0\}$ be the set of arcs that are used in the aggregated solution. If we find a solution with an objective value equal to $|\mathcal{A}^*|$ using this formulation, then the aggregated solution can be disaggregated to a feasible solution for the initial model. Each repetition of the same aggregated arc will be disaggregated as the same arc. We introduce the following decision variables of the problem: for each activity a, u_a represents its starting time. Let $\alpha_a^k, \forall k \in \{1, \ldots, z^*\}, \forall a \in \mathcal{A}^*$, be the binary variable that defines whether activity a is assigned to machine k or not. The binary variable $y_{aa'}$ equals to one if and only if activity a' appears after activity a.

$$\max \quad \sum_{k=1}^{z^*} \sum_{a \in \mathcal{A}^*} \alpha_a^k \tag{5}$$

$$\text{s. t.} \quad \sum_{k=1}^{z^*} \alpha_a^k \leq 1, \qquad \forall a \in \mathcal{A}^*, \tag{6}$$

$$-y_{aa'} - y_{a'a} + \alpha_a^k + \alpha_{a'}^k \geq 1, \qquad \forall a, a' \in \mathcal{A}^*, k = 1, \ldots, z^*, \tag{7}$$

$$u_a + p_a \leq u_{a'} + (W - r_{a'})(1 - y_{aa'}), \qquad \forall a, a' \in \mathcal{A}^*, \tag{8}$$

$$\alpha_a^k \in \{0, 1\}, \qquad \forall a \in \mathcal{A}^*, \quad \forall k = 1, \ldots, z^*, \tag{9}$$

$$y_{aa'} \in \{0, 1\}, \qquad \forall a, a' \in \mathcal{A}^*, \tag{10}$$

$$u_a \in [r_a, d_a], \qquad \forall a \in \mathcal{A}^*. \tag{11}$$

The objective function (5) maximizes the number of activities assigned within their time windows, which is equivalent to minimizing the number of late activities. Constraints (6) ensure that each activity in $\mathcal{A}^*$ is assigned to at most one machine. Moreover, if two jobs a and a' are scheduled on the same machine, either activity a begins after the end of activity a' or the opposite occurs (7), while constraints (8) guarantee that the starting time of the activities are consistent with the $y_{aa'}$ variables (the value $W - r_{a'}$ plays the role of a "big M"). Even though the problem is NP-hard, it can be solved efficiently (depending on the application problem) with the MIP formulation we propose. This is true since the number of arcs in a solution is typically small when compared with the total number of arcs in the original problem,

4.2.2. Heuristic method for the general case

In cases where the number of arcs in $\mathcal{A}^*$ is too large, or if their multiplicity is high, we need a fast constructive heuristic to check the feasibility of a solution. The main idea is to try to build z^* feasible paths with the set of arcs in the solution. If the heuristic is able to build these paths, the solution is feasible. If not, we cannot prove its feasibility.

Treating each unit of flow independently leads to a large pseudo-polynomial number of paths in the solution. Therefore, the proposed heuristic (Algorithm 2) does not build each path individually, but rather builds *paths with multiplicities*. Let A_R be the set of weighted arcs of $\mathcal{A}^*$, each arc a with a residual multiplicity $m(a)$, which is initially equal to x_a. For any path μ under construction, we denote by $t(\mu)$ the head of this path, which corresponds

to the head of the last arc in μ, *i.e.* $t(\mu) = \max\{t(a) : a \in \mu\}$. Let Γ be a set of paths μ, each with a multiplicity $m(\mu)$, equal to the flow going through that path. We will refer to the arcs of μ^j as $a^j_1, \ldots, a^j_{|\mu^j|}$. The arcs from A_R are considered by non-decreasing tails. At each iteration, the first arc a^* from A_R is added to the compatible path with the largest value of head. A path μ^j is compatible with an arc a_i if $t(\mu^j) \in [r_{a_i}, d_{a_i}]$. When an arc is appended to a path, its tail becomes not necessarily the tail it had in the solution, but the smallest possible tail of an equivalent arc in the non-aggregated model. When an arc a is appended to a path μ, the multiplicity of a and μ are updated. If $m(a_i) > m(\mu^j)$, the flow going through a_i in μ^j is $m(\mu^j)$. The multiplicity of a_i is updated to $m(a_i) - m(\mu^j)$ in A_R. If, on the other hand, $m(a_i) \leq m(\mu^j)$, a_i is removed from A_R, and path μ^j is duplicated in two identical paths μ^j and $\mu^{|\Gamma|+1}$ such that $m(\mu^j) = m(a_i)$ and $\mu^{|\Gamma|+1}$ has the residual multiplicity. Then, a_i is appended to path μ^j. If a compatible path does not exist, a new path is created with an initial multiplicity $m(a_i)$ and a_i is removed from A_R. If all arcs are successfully allocated to at most z^* paths, the method returns "true". The complexity of this algorithm is independent of the number of repetitions of the arcs. It only depends on the number of arcs in $\mathcal{A}^*$.

Proposition 4. *The complexity of Algorithm 2 is in* $O(|\mathcal{A}^*|^3)$.

Proof. We first show that $|\Gamma| \leq |\mathcal{A}^*|$. In fact, we show that at any iteration of the algorithm, $|\Gamma| + |A_R| \leq |\mathcal{A}^*|$. At step 1, in any case a unique path $\{a^*\}$ is created, and one arc is removed from A_R. Thus the property is initially true. Assuming that at a given step, $|A_R| \leq |\mathcal{A}^*| - |\Gamma|$, there are three possibilities.

1. a^* is appended to a path μ^{j^*} such that $m(a^*) \geq m(\mu^{j^*})$. In this case, $|\Gamma|$ and $|A_R|$ remain unchanged. Thus the property remains true after this step.
2. a^* is appended to a path μ^{j^*} such that $m(a^*) < m(\mu^{j^*})$. In this case, a^* is removed from A_R, and a new path is created if $m(a^*) < m(\mu^{j^*})$. Therefore, the property remains true.
3. a^* cannot be inserted in any open path and a new path is created. In this case, a^* is removed from A_R and a unique new path is created. Therefore, the property remains true in this case.

Since $|\Gamma| \leq |\mathcal{A}^*|$, after a maximum of $|\mathcal{A}^*|$ iterations, arc a^* is deleted from A_R (either all its repetitions are allocated to existing paths, or a new

path is created). Therefore, after a maximum of $|\mathcal{A}^*|^2$ iterations, all arcs are deleted from A_R and the algorithm stops. Again, since $|\Gamma| \leq |\mathcal{A}^*|$, and since an arc cannot appear twice in a path, each execution of the *while* loop can be executed in $O(|\mathcal{A}^*|)$ (seeking the first compatible path, and copying a path are the two most expensive operations), leading to a total complexity of $O(|\mathcal{A}^*|^3)$. The cost $O(|\mathcal{A}^*| \log |\mathcal{A}^*|)$ for initially sorting the arcs is dominated by this cost. Therefore, the overall complexity of the algorithm is $O(|\mathcal{A}^*|^3)$. □

4.3. Disaggregation schemes

When the model finds an infeasible solution, a disaggregation scheme must be applied. Generally, it consists of adding nodes to the current aggregated multigraph, in such a way that the previous solution is not repeated in the next iteration. Once the disaggregation is performed, the updated model is once again solved, and the whole process is iteratively repeated. These additional nodes correspond to values that do not belong to the current target set as they are aggregated in one of its nodes. Therefore, this procedure is designated as disaggregation.

A simple straightforward way of performing a disaggregation is to do it in a global and regular way. This means that an additional node is considered between every two consecutive nodes of the current scale. The disadvantage of this method is that it does not take into account the solution obtained by the model and may create nodes that are unnecessary. We call this disaggregation procedure **Globally Regular Disaggregation (GRD)**.

Node disaggregation can also be performed locally. This means that the additional nodes to consider originate from the disaggregation of nodes whose aggregation is potentially causing the infeasibility of the current solution. Considering Proposition 3, given the current solution, only nodes that are both tails and heads of arcs used in the solution are considered relevant to disaggregate. This leads to the following procedure, which consists in selecting two of those nodes v and v' that are contiguous in the current set $\mathcal{S}$. Then, given a parameter r, we add to $\mathcal{S}$ the set $\{v + k * r : k \in \mathbb{N}, v < v + k * r < v'\}$. We call this procedure **Locally Regular Disaggregation (LRD)**.

4.4. Convergence

Let $X(G, \ell, u, \mathcal{I})$ denote a flow model. Let $\mathcal{V} = \{0, 1, \ldots, W - 1, W\}$ be the set of nodes of graph G, and $X(\Psi(G), \ell, u, \mathcal{I})$ be the aggregate model

where only the nodes in $\mathcal{S} \subset \mathcal{V}$ are considered. It is trivial that $\Psi^{\mathcal{V}}(G) = G$. In the following, we will consider that we use a *non-degenerate* disaggregation algorithm, *i.e.* if this algorithm is applied to a given set $\mathcal{S}$, it disaggregates at least one element of this set. Note that this can be done without loss of generality, since any disaggregation algorithm can be turned into a non-degenerate one by adding a random element of $\mathcal{V} \setminus \mathcal{S}$ to the output set $\mathcal{S}$.

Theorem 1. *The Iterative Aggregation and Disaggregation Algorithm converges to an optimal solution in a finite number of iterations for any non-degenerate disaggregation scheme.*

Proof. Let k be the current iteration, with x^k the optimal solution of model $X(\Psi^{\mathcal{S}^k}(G), \ell, u, \mathcal{I})$, where $\mathcal{S}^k$ is the current set of nodes. If $\mathcal{S}^k = \mathcal{V}$ or x^k is feasible for model $X(G, \ell, u, \mathcal{I})$, x^k is optimal, since $X(\Psi^{\mathcal{S}^k}(G), \ell, u, \mathcal{I})$ is a relaxation of $X(G, \ell, u, \mathcal{I})$. Thus, the method has converged. Otherwise, the method proceeds to the next iteration, with a consequent addition of $n_k > 0$ new nodes to $\mathcal{S}^k$, in the disaggregation step. Consequently, it takes at most $W - |\mathcal{S}^1|$ iterations to reach an iteration n, where $\mathcal{S}^n = \mathcal{V}$, and thus to convergence. □

5. Applications

As mentioned before, the exact solution algorithm we are proposing can be applied to a variety of problems that may be formulated as our min-cost circulation problem. The aim of our computational experiments is to assess the efficiency of the IADA. To do so, we compare the results of the original network flow model with the results of aggregated models within the frame of our algorithm. We also intend to compare the different approaches for the three main steps of the algorithm (finding an initial scale of aggregated nodes, checking the feasibility of a solution and defining a disaggregation scheme) that were proposed in Section 4. Many combinations of these approaches and their parameters can be used, and the results we present in this section represent a selection of them. This selection was based in some preliminary tests, which are not reported in this paper. The algorithm was implemented in C++ and the network flow model was solved with ILOG CPLEX 12.6. The computational tests were run on a cluster of computers using one Quad-core Intel Xeon E5420 CPU with 2.5GHz and 32GB of RAM.

5.1. Vehicle routing problem (VRP) with time windows and multiple routes

We present some computational results concerning the application of the IADA to the VRP with multiple routes and Time Windows (MVRPTW, see Section 2). We conducted a set of computational experiments on benchmark instances from the literature. The considered instances are those of [18] adapted by [1] for the problem. They are divided into three different groups, according to the distribution of the customers' location, R (randomly generated by a random uniform distribution), C (clustered) or RC (randomly generated and clustered). We take into account the first 40 customers of the considered instances. The maximum duration of a route $t_{\max}$ is fixed to 75 for instances RC and R, and to 220 for instances C. The maximum length of a workday is respectively 960, 1000 and 3390 for RC, R and C.

Furthermore, we consider that the distances between customers are equal to the corresponding Euclidean distances, truncated to two decimal places and then multiplied by 100, to obtain integers. Table 1 reports the computational results obtained for instances with 40 customers. The aggregation function used in our experiments is the conservative function $\Psi_c = (\rho^-, \rho^-)$. Four different values for the initial scaling factor were tested (500, 1000, 1500, 10000). If this value is equal to 500 and $|\mathcal{V}|$ is the number of nodes in the original network, then $|V|/500$ nodes are used in the first aggregation. For each of these values, we tested the three different methods presented in Section 4.1 to compute the initial scale (RIS, ME, MC). As the number of arcs in a solution of the MVRPTW is very small when compared to the total number of arcs in the original problem, the feasibility checking is performed by solving the MIP formulation proposed in Section 4.2.1. Finally, the disaggregation scheme used (Section 4.3) is the local regular disaggregation (LRD), with a number of added nodes equal to 10. The models were run with a time limit of 900 seconds. Columns t of Table 1 report the time, in seconds, required to solve to optimality each of the models. A time equal to 900 seconds represents an instance where optimality was not proved within the defined time limit. Those instances for which the best found solution corresponds to the optimal solution are marked with an (*). Columns *it* of Table 1 report the number of iterations needed to find the optimal solution (including the first). A value equal to one means that no disaggregation phase was performed. Instances for which no feasible solution was found within the time limit are marked with an (-). The last line reports the average time, in seconds, required to solve each group of instances, for those which were solved by all models. The non-aggregated model was tested, but no feasible solution was

found within the time limit of 900 seconds for any of the instances. However, it is important to note that the distances between customers were truncated to two decimal places and then multiplied by 100 to obtain integers, which greatly increases the size of the model. In preliminary tests, the optimality of the solution for the MVRPTW was almost always proved by the feasibility checking, and so we apply the IADA described in Algorithm 1, without using the heuristic version.

Table 1: MVRPTW instances with 40 customers. The values on top of the table correspond with the aggregation factor used (*i.e.* the value by which the initial size is divided). The three methods RIS, ME, MC indicated as those of Section 4.1.

		500						1000						1500						10000					
		RIS		ME		MC		RIS		ME		MC		RIS		ME		MC		RIS		ME		MC	
	val	*it*	*t*	*it*	*t*	*it*	*t*	*it*	*t*	*it*	*t*	*it*	*t*	*it*	*t*	*it*	*t*	*it*	*t*	*it*	*t*	*it*	*t*	*it*	*t*
RC201	1292,16	4	32,0	4	41,9	5	37,3	6	80,3	3	19,0	3	54,2	6	101,7	5	20,2	6	82,3	6	74,9	8	40,8	6	58,8
RC202	1457,89	4	163,0	2	48,9	5	188,8	4	165,1	2	48,6	6	179,4	4	190,1	4	136,0	6	402,4	5	158,1	4	117,4	6	86,7
RC204	1362,34	1	35,7	1	23,5	1	66,8	1	17,2	1	21,1	1	16,1	1	14,8	1	15,0	1	12,7	1	2,8	2	7,6	3	25,1
R203	962,22	2	441,2	1	382,0	2	631,1	2	191,0	1	96,8	2	197,7	2	101,7	2	91,3	2	125,0	3	95,2	3	89,4	3	87,5
R204	858,22	1	900*	1	900*	1	900*	1	249,7	1	900*	1	900*	1	169,2	2	900*	1	134,9	1	23,9	1	14,8	1	12,1
R205	1017,84	2	63,8	1	108,1	2	289,8	5	187,5	2	66,7	5	195,8	4	106,5	5	266,3	4	205,3	5	106,0	4	89,6	5	93,4
R206	927,22	1	77,6	1	157,0	1	307,8	1	32,6	1	59,5	1	64,6	2	76,3	2	112,7	2	82,3	2	41,8	1	6,2	2	32,2
R207	886,22	1	197,6	1	561,2	1	290,4	1	55,1	1	42,3	1	96,2	1	38,4	1	29,2	1	68,1	1	8,9	2	30,5	1	8,7
R208	858,22	1	900*	1	900*	1	900*	1	451,5	1	900*	1	226,5	1	119,0	1	353,5	1	357,3	1	27,2	1	17,0	1	24,8
R209	935,81	4	669,5	-	-	-	-	4	359,6	5	672,8	-	-	6	900*	5	390,5	4	187,8	6	481,4	7	845,4	6	478,7
R210	952,92	1	46,3	1	179,0	1	115,2	2	68,5	1	54,7	2	94,9	3	111,4	3	129,6	3	186,0	3	82,1	4	117,8	5	210,3
R211	869,75	1	379,1	1	900*	1	900*	1	204,0	1	505,0	1	234,5	1	91,9	1	297,9	1	118,6	2	87,0	2	70,5	2	92,0
C201	1168,83	1	15,5	1	21,7	1	19,2	1	6,8	2	22,6	1	8,0	1	4,8	1	5,3	1	4,6	3	16,3	2	6,7	4	12,0
C202	1111,15	1	100,1	1	146,3	1	86,8	1	46,6	1	65,7	1	38,3	1	28,4	1	43,6	1	25,9	3	39,4	2	23,4	3	40,3
C203	1088,55	1	356,1	1	823,8	1	291,1	1	113,2	1	191,5	1	100,3	1	66,0	1	146,7	1	63,4	1	16,0	1	22,6	1	14,4
C204	1039,16	1	900,0	1	-	1	900*	1	378,0	1	845,8	1	440,3	1	246,5	1	613,1	1	190,7	1	36,6	1	24,8	1	38,9
C205	1083,81	1	18,0	1	32,5	1	32,3	1	9,8	1	10,3	1	11,8	1	6,9	1	8,5	1	8,7	1	3,0	1	3,1	1	2,7
C206	1081,37	1	34,5	1	60,9	1	40,3	1	15,5	1	16,7	2	23,9	1	10,8	1	12,4	1	9,3	2	13,5	2	12,3	2	9,9
C207	1055,04	1	216,4	1	300,6	1	844,5	1	51,8	1	93,8	1	149,1	1	27,4	1	53,7	1	39,6	1	10,5	2	22,1	1	7,9
C208	1071,99	1	51,1	1	53,9	1	99,4	1	22,0	1	25,0	1	18,5	1	14,1	1	18,4	1	12,9	1	5,5	1	4,5	1	4,8
Avg.		1,4	139,2	1,2	196,1	1,6	222,7	1,8	109,3	1,3	83,7	1,8	100,6	1,8	71,1	1,9	102,4	1,9	107,7	2,3	45,1	2,4	38,7	2,7	45,8

From Table 1, we can see that, as expected, the average number of iterations always grows for larger values of the initial scale aggregation factor, even though the maximum average number of iterations is 2.8, and the maximum number of iterations for one instance is 8. The average computational times almost always decrease, independently of the method being used (RIS,

ME, or MC). In preliminary computational experiments, RIS almost always performs better for smaller problems, even though it tends to be dominated by the other methods as the scale factor grows. This may mean that as the size of the graph decreases, the more elaborate strategies tend to perform better. On the other hand, in Table 1 none of the three methods clearly dominates.

5.2. Cutting-stock problem

We will now consider the cutting-stock problem (see Section 2). We conducted a set of computational experiments on difficult benchmark instances from recent studies [4], which were also used recently in [7], to survey the best methods for solving the cutting-stock problem. We used the instances called ANI in [7] to perform our expriments. In preliminary tests, the optimality of the solution for the CSP was almost always proved by closing the gap, and so we apply the extended variant of IADA with the heuristic version. The aggregation functions used in our experiments are the conservative function $\Psi_c = (\rho^-, \rho^-)$ and the heuristic function $\Psi_h = (\rho^-, \rho^+)$, where ρ^+ and ρ^- are defined from $\mathcal{V}$ to $\mathcal{S}$ for different values of $|\mathcal{S}|$. The settings are the following: to compute $\mathcal{S}$, we used the Regular Initial Scale (RIS) with parameter values $|\mathcal{V}|/2$, $|\mathcal{V}|/5$, $|\mathcal{V}|/10$, $|\mathcal{V}|/50$, $|\mathcal{V}|/100$, $|\mathcal{V}|/200$, and $|\mathcal{V}|/500$. Globally Regular Disaggregation (GRD) was used with parameter 1 (one point added between each consecutive infeasibilities). The heuristic checker (Algorithm 2) is used. We run each model for a maximum time of one hour. Each class of instances contains 50 instances. Testing the 250 instances with the 8 different versions of the algorithm amounts to almost 2000 hours of computing time. Note that we use the four cores of our processor, whereas only one core was used in [7]. The results are shown in Tables 2 and 3.

Table 2: Number of difficult instances solved in less than 1 hour.

		Aggregation factor (1 = no aggregation)							
n	W	1	2	5	10	50	100	200	500
201	2500	29	50	50	49	43	47	48	49
402	10000	2	1	6	43	33	21	10	3
600	20000	0	0	0	10	39	35	35	10
801	40000	0	0	0	0	33	34	36	20
1002	80000	0	0	0	0	0	27	27	15
Sum / 250		31	51	56	102	148	164	156	97

Table 3: Average time in seconds (rounded down) for solving difficult instances, and average number of iterations. When the method failed to find the optimal solution, we used the time limit 3601 to compute the average value.

		Aggregation factor (1 = no aggregation)							
n	W	1	2	5	10	50	100	200	500
201	2500	1828	367(2.00)	510(2.84)	632(3.88)	902(6,00)	994(7.54)	832(8.48)	951(9.66)
402	10000	3579	3596(1.02)	3395(1.86)	1052(2.12)	1993(4.34)	2755(5.70)	3185(6.92)	3445(7.92)
600	20000	3601	3601(1)	3601(1.02)	3523(1.76)	1936(2.94)	1860(4.08)	2083(5.14)	3217(6.84)
801	40000	3601	3601(1)	3601(1)	3601(1)	2255(2.22)	2097(2.98)	2002(4.02)	2726(5.44)
1002	80000	3601	3601(1)	3601(1)	3601(1)	3601(1.46)	2975(2.04)	2817(2.98)	3138(4.04)
Average		2521,8	2953,2	2941,6	2481,8	2029,4	2136,2	2183,8	2695,4

Our method is able to solve all instances of size 201, 43 instances of size 400 and from half to two-thirds of the instances for larger classes, whereas the non-aggregated arc-flow model is only able to prove optimality for 29 instances of the easiest instances. Note that for instances with 1002 items, without aggregation, in 27 out of 50 cases, the model was too large to fit into the memory of our computer.

As expected, large aggregations lead to a larger number of iterations. For small instances, the trade-off is in favor of small aggregation values. For instances of size 201 and 402, a small aggregation is sufficient to make the model tractable (the best aggregation factors are respectively 2 and 10). For larger instances, it is better to use larger aggregation factors of 100 or 200.

Even for large instances, the upper bound is almost always equal to the value of an optimal solution (when at least one full iteration was performed). For aggregation level 100, our heuristic aggregation leads to an optimal solution for all instances of classes $n = 801$ and $n = 1002$, except for one (for which the process was stopped before the upper bound was computed).

Out of 250 difficult instances, 240 were solved by at least one setting of our program. One instance with $n = 400$ remained unsolved within one hour, three with $n = 600$, one with $n = 800$ and five with $n = 1000$. Obviously, fine parameter tuning and data analysis could lead to better results, but our

goal here is to show that a direct application of our framework provides an optimal solution for much more instances than the original arc-flow model. Actually, this simple setting of our framework is sufficient to produce results that outperform the exact methods compared in [7], and to solve instances whose sizes are beyond the scope of the standard arc-flow model.

5.3. Analysis and comparison of the results for the MVRPTW and CSP

The method to find the initial scale and the disaggregation method that performed better for both applications were respectively RIS (Regular Initial Scale) and LRG (Locally Regular Disaggregation).

For the initial aggregation factor, the computational times for the MVRPTW decrease with its growth, whereas for the CSP the aggregation factor has to be chosen carefully depending on the size of the model. A too small scale will lead to a large number of iterations because symmetries may allow the same non-valid configuration to appear several times. Note also that the computational times tend to be worse for large disaggregation factors. As far as the number of iterations is concerned, it always increases with the aggregation factor of the initial scale for both applications. The most significant difference between the two application problems has to do with the fact that the original arcs of the MVRPTW have less repetitions than the arcs of the CSP. These repetitions are constrained to a smaller part of the graph for the routing problem, due to the time windows. This explains the identified differences in what concerns the methods for the feasibility checking and the optimality proof (with the feasibility checking or the gap closure). The aggregated models always outperform the non-aggregated ones in our experiments. They help to produce tractable models, whereas non-aggregated models generally cannot be solved within the defined time limit (and sometimes cannot even be loaded into the memory).

6. Conclusions

In this paper, we introduce a general framework for solving network flow models through an iterative aggregation and disaggregation scheme. We provide a thorough analysis of the different key aspects underlying this general framework, including the relation between the level of aggregation and the quality of the approximations. The framework proposed in this paper is general and applies to a variety of network flow models. To assess the efficiency of the IADA, we analyze it experimentally against two different rele-

vant problems, a variant of the vehicle routing problem and a cutting stock problem, by comparing the results of the original models with the results of aggregated models within the frame of our algorithm. We propose generic and alternative approaches for the initial aggregation, feasibility checking and disaggregation methods. The results show that even a simple application of our method often produces tractable models for instances where non-aggregated models cannot be solved within the defined time limit, and that it is able to outperform the best methods from the literature.

Acknowledgments

The authors would like to thank the anonymous referees for the useful comments.

The authors would also like to thank José Valério de Carvalho and Boris Detienne for their valuable comments.

Experiments presented in this paper were carried out using the PLAFRIM experimental testbed, being developed under the Inria PlaFRIM development action with support from LaBRI and IMB and other entities: Conseil Régional d'Aquitaine, FeDER, Université de Bordeaux and CNRS (see `https://plafrim.bordeaux.inria.fr/`).

The project was partially funded by FeDER and FCT (FCOMP-01-0124-FEDER-020430).

This work was supported by the Centre National de la Recherche Scientifique (CNRS), by the Campus interdisciplinaire de recherche, d'innovation technologique et de formation Internationale sur la Sécurité et l'Intermodalité des Transports (CISIT), CPER projet ELSAT, and by the Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines (LAMIH) of the Université de Valenciennes et du Hainaut-Cambrésis.

References

[1] Azi, N., Gendreau, M., Potvin, J.-Y., 2007. An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. European Journal of Operational Research 178, 755–766.

[2] Bärmann, A., Liers, F., Martin, A., Merkert, M., Thurner, C., Weninger, D., 2015. Solving network design problems via iterative aggregation. Mathematical Programming Computation 7 (2), 189–217.

[3] Boland, N., Hewitt, M., Marshall, L., Savelsbergh, M., 2015. The continuous time service network design problem. Tech. rep., Optimization on-line.

[4] Caprara, A., Diaz, J. D., Dell'Amico, M., Iori, M., Rizzi, R., 2015. Friendly bin packing instances without integer round-up property. Mathematical Programming Series A and B 150, 5–17.

[5] Christophides, N., Mingozzi, A., Toth, P., 1981. State-space relaxation procedures for the computation of bounds to routing problems. Networks 11, 145–164.

[6] de Carvalho, J. M. V., 2002. LP models for bin packing and cutting stock problems. European Journal of Operational Research 141 (2), 253–273.

[7] Delorme, M., Iori, M., Martello, S., 2016. Bin packing and cutting stock problems: Mathematical models and exact algorithms. European Journal of Operational Research 255 (1), 1–20.

[8] Elhallaoui, I., Metrane, A., Soumis, F., Desaulniers, G., 2010. Multiphase dynamic constraint aggregation for set partitioning type problems. Mathematical Programming 123 (2), 345–370.

[9] Elhallaoui, I., Villeneuve, D., Soumis, F., Desaulniers, G., 2005. Dynamic aggregation of set-partitioning constraints in column generation. Operations Research 53 (4), 632–645.

[10] Garey, M., Johnson, D., 1978. "strong" NP-completeness results: motivation, examples, and implications. Journal of the Association for Computing Machinery 25, 499–508.

[11] Glover, F., 1968. Surrogate constraints. Operations Research 16, 741–749.

[12] Lancia, G., Rinaldi, F., Serafini, P., 2011. A time-indexed LP-based approach for min-sum job-shop problems. Annals of Operations Research 186, 175–198.

[13] Macedo, R., Alves, C., de Carvalho, J. V., 2010. Arc-flow model for the two-dimensional guillotine cutting stock problem. Computers & Operations Research 37 (6), 991 – 1001.

[14] Macedo, R., Alves, C., de Carvalho, J. V., Clautiaux, F., Hanafi, S., 2011. Solving exactly the vehicle routing problem with time windows and multiple routes using a pseudo-polynomial model. European Journal of Operational Research 214 (3), 457–545.

[15] Pessoa, A., Uchoa, E., de Aragão, M., Rodrigues, R., 2010. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. Mathematical Programming Computation 2, 1–32.

[16] Righini, G., Salani, M., May 2008. New dynamic programming algorithms for the resource constrained elementary shortest path problem. Networks 51, 155–170.

[17] Rogers, D. F., Plante, R. D., Wong, R. T., Evans, J. R., 1991. Aggregation and disaggregation techniques and methodology in optimization. Operations Research 39 (4), 553–582.

[18] Solomon, M. M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35 (2), 254–265.

[19] Voge, M.-E., Clautiaux, F., 2012. Theoretical investigation of aggregation in pseudo-polynomial network-flow models. In: et al., A. M. (Ed.), ISCO 2012, 2nd International Symposium on Combinatorial Optimization. Vol. 7422 of LNCS. pp. 213–224.

Algorithm 1: Iterative Aggregation and Disaggregation Algorithm

Input: A network flow multigraph $G = (\mathcal{V}, \mathcal{A}, s, t, e, c)$, a set $\mathcal{I}$, lower and upper bounds ℓ, u

Output: An optimal solution x^* for $X(G, \ell, u, \mathcal{I})$

Compute an initial target set $\mathcal{S}^1 \subseteq \mathcal{V}$;
$optimal \leftarrow False$;
$k \leftarrow 1$;
while *optimal=False* **do**
  Let $\Psi_c^k = (\rho_1^c, \rho_2^c)$ be a conservative aggregation function defined from $\mathcal{V}$ to $\mathcal{S}^k$.
  Solve the aggregated model $X(\Psi_c^k(G), \ell, u, \mathcal{I})$, obtaining solution x;
  Check if x is feasible ;
  if *x is feasible* **then**
    $x^* \leftarrow x$;
    $optimal \leftarrow True$
  if *optimal = False and the heuristic version is used* **then**
    Let $\Psi_h^k = (\rho_1^h, \rho_2^h)$ be a heuristic aggregation function defined from $\mathcal{V}$ to $\mathcal{S}^k$.
    Solve the aggregated model $X(\Psi_c^k(G), \ell, u, \mathcal{I})$, obtaining solution y;
    if *x and y have the same objective value* **then**
      $x^* \leftarrow x$;
      $optimal \leftarrow True$;
  if *optimal = False* **then**
    Apply a disaggregation scheme, obtaining a new node set $\mathcal{S}^{k+1} \subseteq \mathcal{V}$;
    $k \leftarrow k + 1$;
return x^*;

Algorithm 2: Feasibility heuristic

Input: a set of weighted arcs A_R, a value z^* of flow;
Sort the arcs in $A_R = \{a_1, ..., a_{|A_R|}\}$ such that $s(a_i) \leq s(a_{i+1})$;
$\mu^1 \leftarrow \emptyset$; $t(\mu^1) \leftarrow 0$; $m(\mu^1) \leftarrow 0$;
$\Gamma = \{\mu^1\}$;
while $A_R \neq \emptyset$ **do**
    Let a^* be the first element of A_R;
    // find the path compatible with a^* with the maximum head
    Let $j^* = \min \Big\{ j \in \{1, \ldots, |\Gamma|\} : d_{a^*} - p_{a^*} \geq t(\mu^j)$ and $\max\{t(\mu^j), r_{a^*}\} + p_{a^*} \leq W \Big\}$;
    if j^* *exists* **then**
        // a^* is appended to path μ^{j^*}
        if $m(a^*) \geq m(\mu^{j^*})$ **then**
            $m(a^*) \leftarrow m(a^*) - m(\mu^{j^*})$;
        else
            // if the number of repetition of μ^{j^*} is too large, it is split into two paths
            $A_R \leftarrow A_R \setminus \{a^*\}$;
            copy μ^{j^*} into $\mu^{|\Gamma|+1}$;
            $m(\mu^{|\Gamma|+1}) \leftarrow m(\mu^{j^*}) - m(a^*)$;
            $m(\mu^{j^*}) \leftarrow m(a^*)$;
            $\Gamma \leftarrow \Gamma \cup \{\mu^{|\Gamma|+1}\}$;
        $\mu^{j^*} \leftarrow \mu^{j^*} \cup \{a^*\}$;
        $t(\mu^{j^*}) \leftarrow \max\{t(\mu^{j^*}), r_{a^*}\} + p_{a^*}$;
    else
        // a new path is created
        if $\sum_{j=1}^{|\Gamma|} m(\mu^j) + m(a^*) \leq z^*$ **then**
            $A_R \leftarrow A_R \setminus \{a^*\}$;
            $\mu^{|\Gamma|+1} \leftarrow \{a^*\}$; $t(\mu^{|\Gamma|+1}) \leftarrow r_{a^*} + p_{a^*}$; $m(\mu^{|\Gamma|+1}) \leftarrow m(a^*)$;
            $\Gamma \leftarrow \Gamma \cup \{\mu^{|\Gamma|+1}\}$;
        else return *False*;
return *True*