Math. Program., Ser. B (2013) 140:45–76
DOI 10.1007/s10107-012-0626-8

FULL LENGTH PAPER

A stabilized structured Dantzig–Wolfe decomposition method

Antonio Frangioni · Bernard Gendron

Received: 27 March 2010 / Accepted: 3 May 2011 / Published online: 20 December 2012

Abstract We discuss an algorithmic scheme, which we call the stabilized structured Dantzig–Wolfe decomposition method, for solving large-scale structured linear programs. It can be applied when the subproblem of the standard Dantzig–Wolfe approach admits an alternative master model amenable to column generation, other than the standard one in which there is a variable for each of the extreme points and extreme rays of the corresponding polyhedron. Stabilization is achieved by the same techniques developed for the standard Dantzig–Wolfe approach and it is equally useful to improve the performance, as shown by computational results obtained on an application to the multicommodity capacitated network design problem.

Keywords Dantzig–Wolfe decomposition method · Structured linear program · Multicommodity capacitated network design problem · Reformulation · Stabilization

Mathematics Subject Classification (2000) 90C06 · 90C25

This paper is dedicated to Claude Lemaréchal at the occasion of his 65th birthday. Like such a large part of current work in computational nondifferentiable optimization and decomposition methods, our results would have never been possible without his pioneering work.

A. Frangioni (✉)
Dipartimento di Informatica, Università di Pisa,
Largo B. Pontecorvo 3, 56127 Pisa, Italy
e-mail: frangio@di.unipi.it

B. Gendron
Interuniversity Research Centre on Enterprise Networks,
Logistics and Transportation (CIRRELT), Département d'informatique et de recherche opérationnelle,
Université de Montréal, C.P. 6128, Succ. Centre-ville,
Montreal, QC H3C 3J7, Canada
e-mail: Bernard.Gendron@cirrelt.ca

1 Introduction

The Dantzig–Wolfe (DW) decomposition method [14], inspired by an algorithm due to Ford and Fulkerson [16] for optimal multicommodity flow computations, allows dealing efficiently with linear programs (LP) having the following general and common structure:

$$\min_x \left\{ cx : Ax = b \ , \ x \in conv(X) \right\}. \tag{1}$$

Provided that one can efficiently perform linear optimization over the closed set $X \subseteq \mathbb{R}^n$, the idea of DW is that of forming the *Lagrangian relaxation* with respect to the m constraints $Ax = b$ and one generic vector π of *Lagrange multipliers*

$$f(\pi) = \min_x \left\{ L(x, \pi) = cx + \pi(b - Ax) \ : \ x \in X \right\} \tag{2}$$

and solving the corresponding *Lagrangian dual* (equivalent to (1)):

$$\max_\pi \left\{ f(\pi) \ : \ \pi \in \mathbb{R}^m \right\}. \tag{3}$$

DW decomposition solves the Lagrangian dual with Kelley's cutting-plane approach [20,27]. The benefit of DW decomposition is twofold. On the one hand, (1) is usually solved as a relaxation of a difficult mixed-integer linear program (MILP), and under appropriate hypotheses, the lower bound computed by the DW approach is stronger than that of the LP relaxation [3,8–10,21,22]. On the other hand, (2) often decomposes into a number of smaller independent subproblems ($Ax = b$ are *linking* constraints), and the resulting approach can be faster than using ordinary LP technology [10,16, 18,21,25]. This is particularly so if *stabilization* [19] is used to devise variants of the DW approach that are more effective in practice than the non-stabilized cutting-plane approach [6,26].

We aim at improving the DW approach for the cases where $conv(X)$ has an appropriate *structure*: it is a polyhedron whose (possibly "large") description can be "conveniently generated piecemeal" with the information provided by solving (2) for appropriately chosen π (see Assumptions 1–3 below). This is what happens with the standard DW, which is nothing but a *column generation procedure* based on a *reformulation* of $conv(X)$ in a different space of variables, namely the convex (and conical) multipliers that allow to express any point of $conv(X)$ as a convex (conical) combination of its *exponentially many* extreme points (rays). However, different models may be available which have a rather different structure; for our application (see Sect. 2) one has "only" pseudo-polynomially many constraints *and* variables, as opposed to exponentially many variables, but very few constraints, as in the standard DW model.

Under these conditions, it is possible to construct a convergent algorithm that closely mimics the DW approach; this has been done for specific applications (see [21] and the references therein). In this article, we point out that one can develop this idea in a general setting, which we call the *Structured Dantzig–Wolfe (SDW) decomposition approach*. Furthermore, the same stabilization techniques that have shown to be useful for the standard DW can be applied to SDW. This gives rise to *stabilized structured*

Dantzig–Wolfe (S^2DW) algorithms, whose convergence can be analyzed with the help of results in [19], and that can be significantly more efficient in practice. One interesting feature of the S(^{2})DW approach is that the subproblem to be solved at each iteration remains the same; only the master problem changes. Hence, if an implementation of the DW approach is available for one problem for which an alternative model of $conv(X)$ is known, then implementing the S(^{2})DW approach for that problem requires relatively minor changes to the existing code.

The structure of the paper is as follows. Section 2 presents the application motivating our study, the multicommodity capacitated network design problem (MCND), and reviews some of the different forms of decomposition that can be developed for the problem. With the help of the ideas developed for the MCND, the general SDW method is presented in Sect. 3, and its relationships with the original DW method are discussed. Section 4 is devoted to describing how the SDW approach can be stabilized. Section 5 then presents and discusses the results of extensive experiments that demonstrate the computational interest of S^2DW by comparing several decomposition methods for the MCND. Finally, conclusions are drawn in Sect. 6. For the remainder of the paper, $v(\cdot)$ denotes the optimal value of an optimization problem.

2 Decomposition for multicommodity capacitated network design

Given a directed network $G = (N, E)$, where N is the set of nodes and E is the set of arcs, we must satisfy the communication demands between several origin-destination pairs, represented by the set of commodities H. Each $h \in H$ is characterized by a positive communication demand d^h that must flow between the origin s_h and the destination t_h; this is represented by the *deficit vector* $b^h = [b^h_i]_{i \in N}$ with $b^h_i = -1$ if $i = s_h$, $b^h_i = 1$ if $i = t_h$, and $b^h_i = 0$ otherwise. While flowing along an arc (i, j), a communication consumes some of the *arc capacity*, which is originated by installing on the arc any number of *facilities*. Installing one facility on arc $(i, j) \in E$ provides a positive capacity u_{ij} at a nonnegative cost f_{ij}; a routing cost c^h_{ij} also has to be paid for each unit of commodity h moving through (i, j). The problem consists of minimizing the sum of all costs, while satisfying demand requirements and capacity constraints. By defining *flow variables* w^h_{ij}, which represent the fraction of the flow of commodity h on arc $(i, j) \in E$ (i.e., $d^h w^h_{ij}$ is the actual value of the flow), and *design variables* y_{ij}, which define the number of facilities to install on arc (i, j), the MCND can be formulated with the following model, denoted I:

$$\min \sum_{h \in H} \sum_{(i,j) \in E} d^h c^h_{ij} w^h_{ij} + \sum_{(i,j) \in E} f_{ij} y_{ij} \tag{4}$$

$$\sum_{(j,i) \in E} w^h_{ji} - \sum_{(i,j) \in E} w^h_{ij} = b^h_i \qquad i \in N,\ h \in H \tag{5}$$

$$\sum_{h \in H} d^h w^h_{ij} \le u_{ij} y_{ij} \qquad (i, j) \in E \tag{6}$$

$$0 \le w^h_{ij} \le 1 \qquad (i, j) \in E,\ h \in H \tag{7}$$

$$y_{ij} \ge 0 \text{ and integer} \qquad (i, j) \in E \tag{8}$$

The LP relaxation $\bar{I}$ of I, obtained by dropping the integrality requirements in (8), provides rather weak lower bounds (see Sect. 5.3); hence, better formulations are required to provide tighter bounds.

2.1 Dantzig–Wolfe decomposition

There are different ways to apply decomposition techniques to the MCND (see [10] for a related but different problem); here, we focus on the case where the flow conservation equations (5) are relaxed, a choice motivated by the fact that the lower bound computed by the corresponding DW approach improves upon $v(\bar{I})$, the LP relaxation bound. Thus, there is one Lagrange multiplier π_i^h for each $h \in H$ and $i \in N$, and the objective function in (2) is (4) with $d^h c_{ij}^h$ replaced with the Lagrangian cost $\bar{c}_{ij}^h = d^h c_{ij}^h - \pi_i^h + \pi_j^h$ (plus a constant cost $\pi b = \sum_{i \in N} \sum_{h \in H} \pi_i^h b_i^h$). Problem (2) then decomposes into $|E|$ subproblems, one for each arc (i, j):

$$\min \sum_{h \in H} \bar{c}_{ij}^h w_{ij}^h + f_{ij} y_{ij} \tag{9}$$

$$\sum_{h \in H} d^h w_{ij}^h \le u_{ij} y_{ij} \tag{10}$$

$$0 \le w_{ij}^h \le 1 \qquad h \in H \tag{11}$$

$$y_{ij} \ge 0 \text{ and integer} \tag{12}$$

In other words, (2) is "easy" in this case because it is *decomposable*, i.e., $x = [x^k]_{k \in K}$ and $X = \bigotimes_{k \in K} X^k$ for a finite set K, where $K = E$, $k = (i, j)$,

$$X^{ij} = \left\{ x^{ij} = \left[\left[w_{ij}^h \right]_{h \in H} , \; y_{ij} \right] : (10) - (12) \right\}, \tag{13}$$

and each of the disjoint sets X^{ij} has a *single* integer variable. If we relax the integrality constraint (12), then the optimal continuous solution satisfies $y_{ij}^* = \sum_{h \in H} d^h w_{ij}^h / u_{ij}$, the computation of the optimal $[w_{ij}^h]_{h \in H}$ being obtained by solving the LP relaxation of a 0–1 knapsack problem. Now, since the optimal value

$$v_{ij}(y_{ij}) = f_{ij} y_{ij} + \min_w \left\{ \sum_{h \in H} \bar{c}_{ij}^h w_{ij}^h : (10) - (11) \right\}$$

is a convex function of y_{ij} (the partial minimization of a convex function is convex), it is easy to show that the optimal integer solution is either $\lceil y_{ij}^* \rceil$ or $\lfloor y_{ij}^* \rfloor$, whichever provides the best value of $v_{ij}(\cdot)$ (see [2,21] for details). Despite being easy to solve, (9–12) does *not* have the integrality property [20]; thus, DW provides a better bound than the LP relaxation, i.e., $v(1) \ge v(\bar{I})$ and a strict inequality usually holds (see Sect. 5.3). Solving (1) by the DW approach corresponds to considering its *reformulation*

$$\min_\theta \left\{ c\left(\sum_{x\in X} x\theta_x\right) \ : \ A\left(\sum_{x\in X} x\theta_x\right) = b \ , \ \theta \in \Theta \right\} \tag{14}$$

where $\Theta = \{\theta \geq 0 : \sum_{x\in X} \theta_x = 1\}$ is the unitary simplex of proper dimension. Problem (14) can be assumed to be finite-dimensional because only the finite—although extremely large—set of possible extreme optimal solutions $x \in X$ of (2) need to be associated a convex multiplier θ_x. Also, X is compact, for otherwise the extreme rays would also have to be considered. Then, the DW algorithm is just *column generation* in (14): a (small) subset $\mathcal{B} \subseteq X$ is selected, and the *primal master problem*

$$\min_x \left\{ cx : Ax = b \ , \ x \in X_{\mathcal{B}} = conv(\mathcal{B}) \right\} \tag{15}$$

$$\equiv \ \min_\theta \left\{ c\left(\sum_{x\in\mathcal{B}} x\theta_x\right) \ : \ A\left(\sum_{x\in\mathcal{B}} x\theta_x\right) = b \ , \ \theta \in \Theta_{\mathcal{B}} \right\} \tag{16}$$

is solved. This is a *restriction* of (1) where the *inner approximation* $X_{\mathcal{B}}$ is used instead of the original set $conv(X)$. Plugging into (2) the part $\tilde{\pi}$ of the dual optimal solution to (15)/(16) associated to the constraints $Ax = b$ generates a new solution $\bar{x} \in X$. It is easy to prove that either inserting $\bar{x}$ into $\mathcal{B}$ (generating the new variable $\theta_{\bar{x}}$) strictly improves (enlarges) the inner approximation, or the optimal solution to the current (15)/(16) is optimal for (14) $\equiv$ (1); iterating this process with minimal care eventually leads to convergence. In dual terms, the DW method can be described by saying that the *cutting-plane model*

$$f_{\mathcal{B}}(\pi) = \min_x \ \{cx + \pi(b - Ax) \ : \ x \in \mathcal{B}\} \tag{17}$$

is an outer approximation of the dual function, i.e., $f_{\mathcal{B}} \geq f$. Solving (16) corresponds to $f_{\mathcal{B}}$ being minimized instead of f [20]; its optimum $\tilde{\pi}$ is then used as the next point where to evaluate the "true" f, and the corresponding optimal solution $\bar{x}$ to (2) either proves that $\tilde{\pi}$ is optimal for (3), or produces a strictly improved model.

2.2 Disaggregated Dantzig–Wolfe decomposition

It is well-known that in decomposable cases, such as the MCND, a different master problem can be devised. Indeed, $\bar{x}$ optimal for (2) means that $\bar{x} = [\bar{x}^k]_{k\in K}$, where $\bar{x}^k$ is an optimal solution to the kth subproblem. Thus, defining the sets $\mathcal{B}^k = \{\bar{x}^k \ : \ \bar{x} \in \mathcal{B}\}$ one can solve at each iteration a *disaggregated master problem*

$$\min_x \left\{ \sum_{k\in K} c^k x^k \ : \ \sum_{k\in K} A^k x^k = b \ , \ x^k \in X^k_{\mathcal{B}} = conv(\mathcal{B}^k) \ \ k \in K \right\} \tag{18}$$

whose feasible region has the (much) larger $\bar{X}_{\mathcal{B}} = \bigotimes_{k\in K} \ conv(\mathcal{B}^k)$ in place of $conv(\mathcal{B})$. This is written as (16) except that each $x \in \mathcal{B}$ is associated to $|K|$ *independent* convex multipliers θ^k_x, and there are $|K|$ constraints $\sum_{x\in\mathcal{B}} \theta^k_x = 1$, one for each $k \in K$,

instead of one concerning all θs. In other words, each $X^k_{\mathcal{B}}$ is an *independent* inner approximation of $conv(X^k)$, and it is easy to see that, for the same $\mathcal{B} \subseteq X$, $\bar{X}_{\mathcal{B}}$ is a better approximation of $conv(X)$ than $conv(\mathcal{B})$. Note that such a disaggregated DW approach is solving *the same Lagrangian dual* (3) as the aggregated one, as testified by the fact that the "oracle" computing the dual function $f(\pi)$ is exactly the same. The difference is that the disaggregated approach *exploits the (cartesian product) structure of* X to build *a better master problem out of the same oracle information*. The trade-off is that (18) has $|K|$ times more variables than (15)/(16); however, it also has much sparser columns and is often less degenerate. Furthermore, (18) uses the available information more efficiently, which often results in faster convergence, and ultimately in better overall performances [25].

2.3 Binary reformulation and "Structured Decomposition"

An alternative way to solve (1) for the MCND starts with the apparently unrelated definition of a *multiple choice* [11,12] binary formulation of the problem. Since $f_{ij} \geq 0$, we have $y_{ij} \leq \lceil \sum_{h \in H} d^h / u_{ij} \rceil = T_{ij}$ for each arc (i, j). Defining $S_{ij} = \{1, \ldots, T_{ij}\}$, we can introduce two new sets of variables

$$y^s_{ij} \in \{0, 1\} = \begin{cases} 1 & \text{if } y_{ij} = s \\ 0 & \text{otherwise} \end{cases} \quad s \in S_{ij}, \quad (i, j) \in E \tag{19}$$

$$w^{hs}_{ij} \in [0, 1] = \begin{cases} w^h_{ij} & \text{if } y_{ij} = s \\ 0 & \text{otherwise} \end{cases} \quad s \in S_{ij}, \quad (i, j) \in E, \quad h \in H. \tag{20}$$

A binary model for the MCND, which we will denote as $B+$, can be obtained by replacing the w^h_{ij} and y_{ij} variables in (4) and (5) with the new ones by means of the obvious equations

$$y_{ij} = \sum_{s \in S_{ij}} s y^s_{ij} \quad (i, j) \in E \qquad w^h_{ij} = \sum_{s \in S_{ij}} w^{hs}_{ij} \quad (i, j) \in E, \quad h \in H$$

and by replacing (6), (7) and (8) with

$$(s - 1) u_{ij} y^s_{ij} \leq \sum_{h \in H} d^h w^{hs}_{ij} \leq s u_{ij} y^s_{ij} \qquad (i, j) \in E, \quad s \in S_{ij} \tag{21}$$

$$w^{hs}_{ij} \leq y^s_{ij} \qquad (i, j) \in E, \quad h \in H, \quad s \in S_{ij} \tag{22}$$

$$\sum_{s \in S_{ij}} y^s_{ij} \leq 1 \qquad (i, j) \in E \tag{23}$$

Model $B+$ is stronger than model I, since its LP relaxation $\bar{B}+$ satisfies $v(\bar{B}+) \geq v(\bar{I})$. In fact, for any $(i, j) \in E$, (21–23) is a description of $conv(X^{ij})$ [12,21], which implies that $v(\bar{B}+) = v$ (1). While $\bar{B}+$ has a pseudo-polynomial number of

variables and constraints, it can be efficiently generated piecemeal using *the very same Lagrangian relaxation* as in the (disaggregated or not) DW, and following the same basic scheme. The idea is again to construct a restriction of the model where only a few of the y^s_{ij} and w^{hs}_{ij} variables, and the corresponding constraints, are present. This master problem is solved, and, exactly as in DW, the dual optimal variables of the $Ax = b$ constraints are plugged into (2) to generate a new solution $\bar{x}$ $(= [\bar{w}, \bar{y}]) \in X$. The difference, analogous to the difference between the aggregated and the disaggregated DW, lies in how $\bar{x}$ is used. In this case, one simply checks whether the variables y^s_{ij} and w^{hs}_{ij} have already been generated for the specific $s = \bar{y}_{ij}$ and the commodities h such that $\bar{w}^h_{ij} > 0$. If this is true for all $(i, j) \in E$, then the current optimal solution to the master problem is optimal for $\bar{B}+$; otherwise, some of the missing variables and of their corresponding constraints are added to the master problem and the process iterates (see [21] for details). It is clear that the above approach is strongly related to the DW method. In the next section, we describe a general algorithmic scheme that encompasses both, with the aim to develop, in Sect. 4, more efficient *stabilized* versions.

3 The structured Dantzig–Wolfe decomposition method

The SDW method solves problem (1) provided that the same assumptions as in the DW approach hold, and some extra knowledge about the structure of $conv(X)$ is available.

Assumption 1 For a finite vector of variables θ and matrices C, Γ and γ of appropriate dimension, $conv(X) = \{x = C\theta \ : \ \Gamma\theta \leq \gamma\}$.

In other words, one needs a proper *reformulation* of $conv(X)$ in a different space of variables θ. We expect the formulation to be "large," so we require it to be amenable to solution through column generation. This calls for the next two assumptions, which use the following notation: $\mathcal{B} = (\mathcal{B}^c, \mathcal{B}^r)$ is "the bundle", where $\mathcal{B}^c$ is any subset of (the index set of) the variables θ (columns of Γ and C), and $\mathcal{B}^r$ is a subset of (the index set of) the constraints (rows in Γ) which impact *at least one* variable in $\mathcal{B}^c$. Then, $\theta_{\mathcal{B}}$ $[= \theta_{\mathcal{B}^c}]$ is the corresponding subvector of variables, $\Gamma_{\mathcal{B}}$ is the sub-matrix of Γ restricted to the columns in $\mathcal{B}^c$ and the rows in $\mathcal{B}^r$, $\gamma_{\mathcal{B}}$ $[= \gamma_{\mathcal{B}^r}]$ is the corresponding restricted right-hand side, and $C_{\mathcal{B}}$ $[= C^{\mathcal{B}^c}]$ is the restriction of the matrix (linear mapping) C to the columns (variables) in $\mathcal{B}$.

Assumption 2 $\Gamma_{\mathcal{B}}\bar{\theta}_{\mathcal{B}} \leq \gamma_{\mathcal{B}}$ and $\theta = \left[\ \bar{\theta}_{\mathcal{B}}\ ,\ 0\ \right] \Rightarrow \Gamma\theta \leq \gamma$.

Assumption 3 Let $\bar{x}$ be a point such that $\bar{x} \in conv(X)\backslash X_{\mathcal{B}}$; then, it must be easy to update $\mathcal{B}$ and the associated $\Gamma_{\mathcal{B}}$, $\gamma_{\mathcal{B}}$ and $C_{\mathcal{B}}$ to a set $\mathcal{B}' \supset \mathcal{B}$ (which satisfies Assumption 2) such that there exists $\mathcal{B}'' \supseteq \mathcal{B}'$ with $\bar{x} \in X_{\mathcal{B}''}$.

Assumption 2 means that we can always "pad" a partial solution with zeroes without losing feasibility. Note that this assumption is automatically satisfied if $\mathcal{B}^r$ contains *all* rows which impact (at least one of) the variables in $\mathcal{B}^c$; this is what happens in the DW approach, where there is only one constraint, or at most one for each component

⟨ initialize $\mathcal{B}$ ⟩;
repeat
  ⟨ solve (25) for $\tilde{x}$; let $\tilde{v} = c\tilde{x}$ and $\tilde{\pi}$ be optimal dual multipliers of $Ax = b$ ⟩;
  $\bar{x} \in \operatorname{argmin}_x \{ (c - \tilde{\pi}A)x : x \in X \}$;
  ⟨ update $\mathcal{B}$ as in Assumption 3 ⟩;
until $\tilde{v} \leq c\bar{x} + \tilde{\pi}(b - A\bar{x})$

Fig. 1 The structured Dantzig–Wolfe algorithm

in the disaggregated case. In general, however, one does not want to impose such a requirement, since the variables can appear in "many" constraints, and one does not want to generate them all. For instance, in the MCND, each variable y^s_{ij} is involved in many constraints (22), and one wants to avoid to generate them all, as much as to avoid generating all the corresponding flow variables w^{hs}_{ij}. Actually, one could also avoid to generate some constraints in $\mathcal{B}^r$ that are nominally necessary to obtain Assumption 2, provided that when solving the master problem (see 25 below), one checks for their violation, adds the violated constraints and re-solves. This is in fact done in [21] for constraints (22). However, this is only a technical detail, as these constraints can be counted as "present" in $\mathcal{B}^r$. The crucial fact is that Assumption 2 (which clearly holds for the MCND) implies

$$\begin{aligned} &X_{\mathcal{B}} = \left\{x = C_{\mathcal{B}}\theta_{\mathcal{B}} \ : \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}\right\} \ \subseteq \ conv(X) \quad \text{and therefore} \\ &f_{\mathcal{B}}(\pi) = \min_x \{cx + \pi(b - Ax) \ : \ x = C_{\mathcal{B}}\theta_{\mathcal{B}}, \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}\} \geq f, \end{aligned} \tag{24}$$

i.e., $f_{\mathcal{B}}$ is a model of f. In fact, (24) reduces to (17) when $X_{\mathcal{B}}$ is defined as in (15)/(16). This justifies the name "bundle" for $\mathcal{B}$, since it can still be interpreted as a subset of the whole information required to describe the behavior of f. Assumption 3 then requires that, given a point $\bar{x} \in conv(X)$ that *cannot* be expressed by means of a given $\mathcal{B}$, it must be easy to update $\mathcal{B}$ ($C_{\mathcal{B}}$, $\Gamma_{\mathcal{B}}$ and $\gamma_{\mathcal{B}}$) in order to incorporate at least some of the information provided by $\bar{x}$. This is purposely stated in a rather abstract form: one just has to find out "why" $\bar{x}$ is not feasible, that is, to find at least one variable that is not in $\theta_{\mathcal{B}}$ and that is needed to represent it. However, Assumption 3 does *not* require $\bar{x} \in X_{\mathcal{B}'}$, i.e., that enough variables are added to "capture" $\bar{x}$. This might require "many" new variables (and constraints), but $\bar{x}$ is unlikely to be the optimal solution anyway. For the algorithm to work, one can as well be content with inserting in $\mathcal{B}$ *at least one* of the "missing" variables; eventually, all the required ones will be generated.

Under Assumptions 1–3 above, the *primal master problem* (in "explicit" form)

$$\min_{x,\theta} \left\{cx \ : \ Ax = b, \ x = C_{\mathcal{B}}\theta_{\mathcal{B}}, \ \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \leq \gamma_{\mathcal{B}}\right\} \tag{25}$$

has the same "implicit form" (15) as in the standard DW, and can be similarly updated once (2) is solved; indeed, (25) reduces to (15)/(16) when $X_{\mathcal{B}} = conv(\mathcal{B})$. Thus, the *structured Dantzig–Wolfe decomposition method* (SDW), as described by the pseudo-code of Figure 1, is "just DW with (25) in place of (16)."

It is reasonably easy to show that, under appropriate assumptions, the algorithm is correct and finitely terminates; this is due to the fact that at each iteration where the

algorithm does not stop, "new" information is added to $\mathcal{B}$, and $conv(X)$ only needs a finite $\mathcal{B}$ to be completely described. Termination can be proven even if variables and constraints are (carefully) removed from $\mathcal{B}$. Furthermore, X does not need to be compact as apparently required by the above notation. This requires that the master problem (25) itself is not unbounded below, i.e., $(\tilde{x}, \tilde{\pi})$ exist in the first place, and then a strengthened form of Assumption 3 to handle the case where $f(\tilde{\pi}) = -\infty$ (see the Appendix). We do not dwell further upon convergence of SDW, since is it basically subsumed by the convergence theory discussed in Sect. 4 for its *stabilized* versions.

The basic ingredients of the SDW approach are the same as in DW: *reformulation* and *column generation*. For DW, however, the reformulation part is standard, and therefore tends to be overlooked; this is not the case for SDW, where one exploits information about the structure of $conv(X)$ to construct a better master problem out of the same information. Clearly, the disaggregated DW (see Sect. 2.2) is the special case of this approach for the very widespread (and simple) block-separable structure. It is also useful to remark that similar ideas have been proposed in the *column generation* community, where it is well-known that the master problem must have very sparse columns (8–12 nonzeroes are usually considered the limit; see, for instance, [15]) in order to avoid degeneracy, and therefore attain good performance. Hence, it is usually beneficial to replace one dense column with several sparse ones. Examples of this idea include replacing a single tree with all the paths from the root to the leaves in multicommodity flows [25], and replacing long paths by shorter ones linked by flow conservation constraints in the master problem of vehicle routing problems [29]. Another related approach is that of dual-optimal inequalities, described below. SDW goes further in the direction of exploiting the data generated by solving (2) "at a finer granularity" than just making each solution a column. This is confirmed by the fact that, for the MCND, columns in the master problem of the standard DW can have as many as $2|N||H| + 1$ nonzeroes (two for each $w^h_{ij} > 0$, corresponding to the flow conservation constraints of i and j, plus one in the convexity constraint), while those in the disaggregated DW can have at most $2|H| + 1$ nonzeroes. The y^s_{ij} columns in the master problem of SDW have no more than $|H| + 2$ nonzeroes, and the w^{hs}_{ij} columns have exactly 5 nonzeroes; thus SDW has significantly sparser columns than DW, disaggregated or not.

Given the availability of efficient and well-engineered linear optimization software, SDW is not significantly more difficult to implement than DW. However, as for the latter, several nontrivial issues have to be addressed.

Initialization. In order to be well-defined, the SDW method needs a starting $\mathcal{B}$ such that (25) has a solution. From the dual viewpoint, the infeasibility of (25) means that no optimal multipliers $\tilde{\pi}$ exist: the dual is unbounded. Avoiding this is in general nontrivial, and requires something akin a "Phase 0" approach where

$$\min_x \left\{ ||Ax - b|| \ : \ x \in X_{\mathcal{B}} \right\}$$

(for a proper norm) is solved at each iteration instead of (25) until a $\mathcal{B}$ which provides a feasible solution is found, or (1) is proved to be infeasible. This can sometimes be avoided by exploiting the structure of X: for the MCND, one can simply find the

optimal solution of $\bar{I}$ (the LP relaxation of formulation 4–8) and initialize $\mathcal{B}$ in such a way that at least that solution is feasible for (25).

Instability. As in the standard DW case, the sequence of dual solutions $\{\tilde{\pi}\}$ cannot be expected to have—and does not have, in practice—good "locality" properties: even if a good approximation of the dual optimal solution π^* is obtained at some iteration, the dual solution at the subsequent iteration may be arbitrarily far from optimal (see, for instance [6,20,27]). This instability is one of the main causes of its slow convergence rate on many practical problems.

All this suggests to introduce some mean to "stabilize" the sequence of dual iterations, exploiting ideas originally developed in the field of nondifferentiable optimization [19] and recently applied to column generation [6]. It is interesting to remark that, while perhaps the most straightforward, that approach—described in the next Section—is not the only possible one. An alternative is that of *dual-optimal inequalities* [1,7], that is, inequalities in the dual space (columns in the primal) cutting away parts of the feasible region of (3) without eliminating all dual optimal solutions. The primal interpretation of this process is particularly relevant for our discussion: what one does is to *reformulate* the model adding "useless" columns. For instance, in cutting stock problems [7] dual-optimal inequalities correspond to columns which basically state that each item i in a cutting pattern can be replaced by a given subset of items whose combined weight is not larger than that of i. In multicommodity flow problems [1], dual-optimal inequalities are cycles which allow to redirect the (aggregated) flow from one arc along a different path. These columns are not necessary to build an optimal solution, *if* all "normal" columns are present. However, in a master problem where the set of columns is by definition (severely) restricted, the addition of a small set of these columns allow to greatly increase the part of the feasible space of the original problem that can be represented, thereby (hopefully) allowing to generate the optimal solution much faster. Again, the concept is that it can be beneficial to modify the master problem in order to allow the information carried by a given set of "normal" columns to be "mixed and matched more freely" than it is possible in the standard DW approach; the increase in the size of the master problem can be largely compensated by the improvement in the convergence speed (note that dual-optimal columns are usually very sparse as well).

4 Stabilizing structured Dantzig–Wolfe decomposition

In order to avoid large fluctuations of the dual multipliers, a "stabilization device" is introduced in the dual problem. This is done by choosing a *current center* $\bar{\pi}$, a family of proper convex *stabilizing functions* $\mathcal{D}_t : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ dependent on a real parameter $t > 0$, and by solving the *stabilized dual master problem*

$$\max_{\pi} \left\{ f_{\mathcal{B}}(\pi) - \mathcal{D}_t(\pi - \bar{\pi}) \right\} \tag{26}$$

at each iteration. The optimal solution $\tilde{\pi}$ of (26) is then used to compute $f(\tilde{\pi})$ as in the standard scheme. The stabilizing function $\mathcal{D}_t$ is meant to penalize points "too far" from $\bar{\pi}$; at a first reading a norm-like function can be imagined there, with more

details to be given soon. Other ways to stabilize the cutting-plane algorithm have been proposed; a thorough discussion of the relationships among them can be found in [24,27].

Problem (26) is a *generalized augmented Lagrangian* of (15), using as augmenting function the *Fenchel conjugate* of $\mathcal{D}_t$. For any convex function $\psi(\pi)$, its Fenchel conjugate $\psi^*(z) = \sup_\pi \{z\pi - \psi(\pi)\}$ characterizes the set of all vectors z that are support hyperplanes to the epigraph of ψ at some point. The function ψ^* is closed and convex, and has several other properties [19,24]; here, we just recall that, from the definition, $\psi^*(0) = -\inf_\pi \{\psi(\pi)\}$. In our case, we obtain the well-known result that *the Fenchel conjugate of a dual function is the value function of the original problem*. In fact,

$$\begin{aligned}(-f_{\mathcal{B}})^*(z) &= \sup_\pi \{z\pi + f_{\mathcal{B}}(\pi)\} \\ &= \sup_\pi \left\{\min_x \{cx + \pi(z + b - Ax) \ : \ x \in X_{\mathcal{B}}\}\right\} \qquad (27) \\ &= \min_x \left\{cx \ : \ z = Ax - b, \ x \in X_{\mathcal{B}}\right\} \qquad (28)\end{aligned}$$

where the last equality comes from standard results in Lagrangian duality [24]: (27) is the Lagrangian dual of (28) with respect to the "perturbed" constraints $z = Ax - b$. We can then compute the *Fenchel dual* [24] of (26), which is obtained by simply writing down $(-f_{\mathcal{B}})^*(0)$; after some algebra, this can be shown to result in

$$\min_z \left\{(-f_{\mathcal{B}})^*(z) - \bar{\pi} z + \mathcal{D}_t^*(-z)\right\}$$

which, plugging in the definition of $f_{\mathcal{B}}^*$, gives

$$\min_{z,x} \left\{cx - \bar{\pi} z + \mathcal{D}_t^*(-z) \ : \ z = Ax - b, \ x \in X_{\mathcal{B}}\right\}. \qquad (29)$$

The *stabilized primal master problem* (29) is equivalent to (26); indeed, we can assume that whichever of the two is solved, both an optimal primal solution $(\tilde{x}, \tilde{z})$ and an optimal dual solution $\tilde{\pi}$ are simultaneously computed. For practical purposes, one would more likely implement (29) rather than (26); however, the following proposition shows how to recover all the necessary dual information once (29) is solved.

Theorem 4 *Let $(\tilde{x}, \tilde{z})$ be an optimal primal solution to (29) and $\tilde{\pi}$ be optimal Lagrange (dual) multipliers associated to constraints $z = Ax - b$; then, $\tilde{\pi}$ is an optimal solution to (26), and $f_{\mathcal{B}}(\tilde{\pi}) = c\tilde{x} + \tilde{\pi}(b - A\tilde{x})$.*

Proof Because $\tilde{\pi}$ are optimal dual multipliers, $(\tilde{x}, \tilde{z})$ is an optimal solution to

$$\min_{z,x} \left\{cx - \bar{\pi} z + \mathcal{D}_t^*(-z) + \tilde{\pi}(z - Ax + b) \ : \ x \in X_{\mathcal{B}}\right\}.$$

That is, $\tilde{z} \in \operatorname{argmin}_z\{(\tilde{\pi} - \bar{\pi})z + \mathcal{D}_t^*(-z)\}$, which is equivalent to $0 \in \partial[(\tilde{\pi} - \bar{\pi}) \cdot + \mathcal{D}_t^*(-\cdot)](\tilde{z})$, which translates to $0 \in \{\tilde{\pi} - \bar{\pi}\} - \partial\mathcal{D}_t^*(-\tilde{z})$, which finally yields $\tilde{\pi} - \bar{\pi} \in \partial\mathcal{D}_t^*(-\tilde{z})$. Furthermore, since $\tilde{x} \in \operatorname{argmin}_x\{cx + \tilde{\pi}(b - Ax) \ : \ x \in X_{\mathcal{B}}\}$, one first has $f_{\mathcal{B}}(\tilde{\pi}) = c\tilde{x} + \tilde{\pi}(b - A\tilde{x})$ as desired. Then, because $\tilde{x}$ is

⟨ Initialize $\bar{\pi}$ and t ; solve $P_{\bar{\pi}}$, initialize $\mathcal{B}$ with the resulting $\bar{x}$ ⟩
repeat
  ⟨ solve (29) for $\tilde{x}$; let $\tilde{\pi}$ be optimal dual multipliers of $z = Ax - b$ ⟩;
  if($c\tilde{x} = f(\bar{\pi})$ & $A\tilde{x} = b$) **then STOP**;
  else $\bar{x} \in \operatorname{argmin}_x \{ (c - \tilde{\pi}A)x : x \in X \}$; $f(\tilde{\pi}) = c\bar{x} + \tilde{\pi}(b - A\bar{x})$;
    ⟨ update $\mathcal{B}$ as in Assumption 3 ⟩;
    if($f(\tilde{\pi})$ is "substantially better" than $f(\bar{\pi})$)
    then $\bar{\pi} = \tilde{\pi}$
    ⟨ possibly update t ⟩
until STOP

Fig. 2 The stabilized structured Dantzig–Wolfe algorithm

a minimizer, $b - A\tilde{x}$ is a *supergradient* of the *concave* function $f_{\mathcal{B}}$; changing sign, $-(b - A\tilde{x}) = \tilde{z} \in \partial(-f_{\mathcal{B}})(\tilde{\pi})$. The desired results now follow from [19, Lemma 2.2, conditions (2.2) and (2.3)] after minor notational adjustments ($-f_{\mathcal{B}}$ in place of $f_{\mathcal{B}}$, $\tilde{\pi} - \bar{\pi}$ in place of d^*). □

Using $\mathcal{D}_t = \frac{1}{2t}\|\cdot\|_2^2$, which gives $\mathcal{D}_t^* = \frac{1}{2}t\|\cdot\|_2^2$, one immediately recognizes in (29) the *augmented Lagrangian* of (15), with a "first-order" Lagrangian term corresponding to the current point $\bar{\pi}$ and a "second-order" term corresponding to the stabilizing function $\mathcal{D}_t$. Using a different stabilizing term $\mathcal{D}_t$ in the dual corresponds to a non-quadratic augmented Lagrangian. The "null" stabilizing term $\mathcal{D}_t = 0$ corresponds to $\mathcal{D}_t^* = I_{\{0\}}$; that is, with no stabilization at all, (29) collapses back to (15)/(25). This is the extreme case of a general property, that can be easily checked for varying t in the quadratic case; as $f \leq g \Rightarrow f^* \geq g^*$, a "flatter" $\mathcal{D}_t$ in the dual corresponds to a "steeper" $\mathcal{D}_t^*$ in the primal, and vice-versa. Also, note that the above formulae work for $X_{\mathcal{B}} = X$ as well, i.e., for the original problems (1)/(3) rather than for their approximations.

The stabilized master problems provide means for defining a general stabilized structured Dantzig–Wolfe algorithm (S^2DW), such as that of Fig. 2.

The algorithm generates at each iteration a *tentative point* $\tilde{\pi}$ for the dual and a (possibly infeasible) primal solution $\tilde{x}$ by solving (29). If $\tilde{x}$ is feasible and has a cost equal to the lower bound $f(\bar{\pi})$, then $\tilde{x}$ and $\bar{\pi}$ are clearly optimal for (1) and (3), respectively. In practice, one can stop the algorithm when $c\tilde{x} + \tilde{\pi}(b - A\tilde{x}) - f(\bar{\pi}) \geq 0$ and $||A\tilde{x} - b||$ (with any norm) are both "small" numbers; we use

$$c\tilde{x} + \tilde{\pi}(b - A\tilde{x}) - f(\bar{\pi}) + t^*||A\tilde{x} - b|| \leq \varepsilon f(\bar{\pi}) \tag{30}$$

where the norm is chosen to match the stabilizing term (see Sect. 4.2), $\varepsilon > 0$ is the *relative* accuracy required to the function value (`1e-6` in our tests), and t^* is an appropriately chosen factor, depending on the scaling of the dual function f, that weights the relative contribution of the constraints violation. The rationale of (30) is that it is usually fairly easy to find a value for t^* which is both "reasonably small" and correct, in the sense that $\bar{\pi}$ is actually ε-optimal at termination; this value is typically quite "stable" within instances of (1)/(3) generated in the same way.

If (30) is not satisfied, new elements of $\mathcal{B}$ are generated by using the primal solution $\bar{x}$ obtained by solving (2) at the tentative point $\tilde{\pi}$. If $f(\tilde{\pi})$ is "substantially better" than $f(\bar{\pi})$, then it is worth to update the current center; this is called a *Serious Step (SS)*. Otherwise, the current center is not changed, and we rely on the fact that $\mathcal{B}$ is improved for producing, at the next iteration, a better tentative point $\tilde{\pi}$; this is called a *Null Step (NS)*. In either case, the stabilizing term can be changed, usually in different ways according to the outcome of the iteration. If an *SS* is performed, i.e., the current approximation $f_{\mathcal{B}}$ of f was able to identify a point $\tilde{\pi}$ with better function value than $\bar{\pi}$, then it may be worth to "trust" the model more and lessen the penalty for moving far from $\bar{\pi}$; this corresponds to a "steeper" penalty term in the primal. Conversely, a "bad" *NS* might be due to an excessive trust in the model, i.e., an insufficient stabilization, thereby suggesting to "steepen" $\mathcal{D}_t$ ($\Rightarrow$ "flatten" $\mathcal{D}_t^*$).

When $f_{\mathcal{B}}$ is the standard cutting-plane model (17), the above approach is exactly a (generalized) bundle method [19]; thus, S^{2}DW is a bundle method where the model $f_{\mathcal{B}}$ is "nonstandard." Note that the bundle method exists in the disaggregated variant for the decomposable case using the stabilized version of (18) as master problem [3,9]; already in that case $f_{\mathcal{B}}$ is not the cutting-plane model, but rather the sum of $|K|$ independent cutting-plane models, one for each independent Lagrangian function. S^{2}DW can be seen as only carrying this idea to its natural extension by using an "even more disaggregated" (specialized) model in place of the standard cutting-plane one(s).

4.1 Convergence conditions

The S^{2}DW algorithm can be shown to finitely converge to a pair (π^*, x^*) of optimal solutions to (3) and (1), respectively, under the following conditions:

(i) $\mathcal{D}_t$ is a convex nonnegative function such that $\mathcal{D}_t(0) = 0$, its level sets $S_\delta(\mathcal{D}_t)$ are *compact and full-dimensional* for all $\delta > 0$; remarkably, these requirements are *symmetric* in the primal, i.e., they hold for $\mathcal{D}_t^*$ if and only if they hold for $\mathcal{D}_t$ [19].

(ii) $\mathcal{D}_t$ is *differentiable* in 0 and *strongly coercive*, i.e., $\lim_{\|\pi\|\to\infty} D_t(\pi)/\|\pi\| = +\infty$; equivalently, $\mathcal{D}_t^*$ is *strictly convex* in 0 and *finite everywhere*.

(iii) For some fixed $m \in (0, 1]$, the condition

$$f(\tilde{\pi}) - f(\bar{\pi}) \geq m(f_{\mathcal{B}}(\tilde{\pi}) - f(\bar{\pi})). \tag{31}$$

is necessary for an SS to be declared. The condition is also sufficient at length; that is, one can avoid to perform an SS if (31) holds, but only finitely many times.

(iv) During an infinite sequence of *consecutive NSs*, $f(\tilde{\pi})$ must be computed and $\mathcal{B}$ updated as in Assumption 3 infinitely many times.

(v) t is bounded away from zero ($t \geq \underline{t} > 0$), and during a sequence of *consecutive NSs*, t can change only *finitely many times*.

(vi) D_t is nonincreasing as a function of t, and $\lim_{t\to\infty} D_t(\pi) = 0$ for all π, i.e., it converges pointwise to the constant zero function. Dually, D_t^* is nondecreasing as a function of t and converges pointwise to $I_{\{0\}}$.

Under the above assumptions, global convergence of S^{2}DW can be proven, mostly relying on the results of [19]. The only delicate point is the treatment of $\mathcal{B}$ along the iterations. In fact, the theory in [19] does not mandate any specific choice for the model apart from $f_{\mathcal{B}} \geq f$, thereby allowing the use of (24). However, the handling of $\mathcal{B}$ (called "β-strategy" in [19]) requires some care. The basic requirement is that it is *monotone* [19, Definition 4.6]: this means that *at length, during a sequence of consecutive NSs*

$$(-f_{\mathcal{B}+})^*(\tilde{z}) \leq (-f_{\mathcal{B}})^*(\tilde{z}) \tag{32}$$

where $\mathcal{B}+$ is the bundle at the subsequent iteration. From (28), it is clear that (32) can be obtained with some sort of monotonicity in $\mathcal{B}$; trivially, by never removing anything. One can do better, allowing removals from $\mathcal{B}$ as long as "the important variables" are left in; this is stated in the following Lemma, whose proof is obvious.

Lemma 1 *A sufficient condition for (32) to hold is that (for any sequence of NSs, at length) the bundle $\mathcal{B}$ has the following property: the optimal solution $\tilde{x}$ to (29) at any step is still feasible for (29) at the subsequent step, i.e., with bundle $\mathcal{B}+$.*

Hence it is only necessary to look, at each iteration, at the optimal value $\theta^*_{\mathcal{B}}$ of the "auxiliary" variables in the current definition of $X_{\mathcal{B}}$; all the variables with zero value can be discarded. However, convergence requires more than just non-descent, which in turn requires a slightly "stronger grip" on $\mathcal{B}$ (see Lemma 2). Furthermore, there is one minor but noteworthy aspect that does not allow to use the results of [19] directly. Indeed, a significant part of the convergence of bundle methods hinges on the assumption that "frequently enough," once f is computed at the trial point $\tilde{\pi}$, the model $f_{\mathcal{B}+}$ at the following iteration must take into account the corresponding subgradient $b - A\bar{x} = \bar{z} \in \partial f(\tilde{\pi})$, in the sense that $f^*_{\mathcal{B}+}(\bar{z}) \leq f^*(\bar{z})$ [19, (4.iv)]. In the standard DW case, this is easily obtained by adding $\bar{z}$ to $\mathcal{B}$, but in our case, this is not, in general, true. Indeed, in accordance with Assumption 3, (iv) above is weaker than [19, (4.iv)], in that it is not required for the model at the next iteration to be capable of *entirely* representing $\bar{x}$ (and hence $\bar{z}$), but only to be "at least a little bit larger." This is discussed in some details in the Appendix, where a sketch of convergence results for the method is presented. The concept is discussed here in order to be able to bring about a related interesting point: that of *aggregation*. In general, monotonicity only requires that the optimal solution $(\tilde{x}, \tilde{z})$ to (29) remains feasible at the subsequent iteration. When using the standard cutting-plane model (17), there is a particularly simple way of attaining this: it suffices to *add $\tilde{x}$ to $\mathcal{B}$*, possibly *removing every other point*. Indeed, the linear function $f_{\tilde{x}}(\pi) = c\tilde{x} + \pi(b - A\tilde{x})$ is a model of f, since $\tilde{x} \in X_{\mathcal{B}} \subseteq X$, and therefore $\tilde{z} = b - A\tilde{x}$ is feasible to (29). While such a harsh approximation of f is contrary in spirit to the S^{2}DW approach, it is worth remarking that under mild conditions, performing aggregation is possible even with a different model than the cutting-plane one. The idea is to consider $\bar{f}_{\mathcal{B}} = \min\{f_{\mathcal{B}}, f_{\tilde{x}}\}$, where $f_{\mathcal{B}}$ is (24); this is clearly a model, as $f \leq f_{\tilde{x}}$ and $f \leq f_{\mathcal{B}}$ imply $f \leq \bar{f}_{\mathcal{B}}$, and $\bar{f}_{\mathcal{B}}$ is "at least as accurate" as $f_{\mathcal{B}}$. A little conjugacy calculus [24] then gives

$$epi\ \left(-\bar{f}_{\mathcal{B}}\right)^* = cl\ conv\big(epi\ (-f_{\mathcal{B}})^*,\ epi\ (-f_{\tilde{x}})^*\big),$$

where it is easy to verify that $(-f_{\tilde{x}})^*(z) = c\tilde{x}$ if $z = A\tilde{x} - b$, and $(-f_{\tilde{x}})^*(z) = +\infty$ otherwise. Thus, the epigraph of $(-\bar{f}_{\mathcal{B}})^*$ can be constructed by taking all points (in the epigraphical space) $(z', (-f_{\mathcal{B}})^*(z'))$ and computing their convex hull with the single point $(A\tilde{x} - b, c\tilde{x})$, as $z'' = A\tilde{x} - b$ is the only point at which $(-f_{\tilde{x}})^*$ is not $+\infty$. The function value is then the inf over all these possibilities for a fixed z, i.e.,

$$\left(-\bar{f}_{\mathcal{B}}\right)^*(z) = \min_{z',\rho}\left\{\rho(-f_{\mathcal{B}})^*(z') + (1-\rho)c\tilde{x} : z = \rho z' + (1-\rho)(A\tilde{x} - b), \rho \in [0,1]\right\}.$$

Therefore, the stabilized primal master problem (29) using model $\bar{f}_{\mathcal{B}}$ is

$$\min_{z,z',x',\theta'_{\mathcal{B}},\rho} \begin{cases} \rho c x' + (1-\rho)c\tilde{x} - \bar{\pi} z + \mathcal{D}_t^*(-z) \\ z' = Ax' - b, \;\; x' = C_{\mathcal{B}}\theta'_{\mathcal{B}}, \;\; \Gamma_{\mathcal{B}}\theta'_{\mathcal{B}} \le \gamma_{\mathcal{B}} \\ z = \rho z' + (1-\rho)(A\tilde{x} - b), \;\; \rho \in [0,1]. \end{cases}$$

We can apply simple algebra to eliminate z', but this still leaves ungainly bilinear terms $\rho x'$ in the problem (the terms $(1-\rho)c\tilde{x}$ and $(1-\rho)(A\tilde{x} - b)$ pose no problem instead, as $\tilde{x}$ is a constant). However, *provided that* $\{\, \theta_{\mathcal{B}} \;:\; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \le \gamma_{\mathcal{B}} \,\}$ *is compact*, these can be effectively eliminated by the variable changes $x = \rho x'$ and $\theta_{\mathcal{B}} = \rho\theta'_{\mathcal{B}}$, which leads to

$$\min_{z,x,\theta_{\mathcal{B}},\rho} \begin{cases} cx + (1-\rho)c\tilde{x} - \bar{\pi} z + \mathcal{D}_t^*(-z) \\ x = C_{\mathcal{B}}\theta_{\mathcal{B}}, \;\; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \le \rho\gamma_{\mathcal{B}} \\ z = Ax + (1-\rho)A\tilde{x} - b, \;\; \rho \in [0,1]. \end{cases} \tag{33}$$

The problems are clearly equivalent: from the compactness assumption, $\{\theta_{\mathcal{B}} \;:\; \Gamma_{\mathcal{B}}\theta_{\mathcal{B}} \le 0\} = \{0\}$, and therefore $\rho = 0 \Rightarrow \theta_{\mathcal{B}} = 0 \Rightarrow x = 0$ as expected. When $\rho > 0$ instead, one simply has $x' = x/\rho$ and $\theta'_{\mathcal{B}} = \theta_{\mathcal{B}}/\rho$, and the equivalence follows algebraically. The master problem therefore needs very little and intuitive modification: the new variable ρ is a "knob" that allows either picking the fixed solution $\tilde{x}$ ($\rho = 0$), or any solution in $X_{\mathcal{B}}$ ($\rho = 1$), or "anything in between". Note that, for the purpose of finite termination, performing aggregations is dangerous: if $\tilde{x}$ is allowed to change at each iteration, the set of all possible models $\bar{f}_{\mathcal{B}}$ is *not* finite. Thus, one has (at least in theory) to resort to something like a *safe* β-strategy [19, Definition 4.9], which is simply one where the total number of aggregations is finite (however this is obtained).

Convergence can also be obtained under weaker conditions. For instance:

- Strong coercivity in (ii) can be relaxed provided that, as in the non-stabilized case, $\mathcal{B}$ is "sufficiently large" to ensure that (29) has at least one feasible solution. This actually provides a possible use for (33) in case one knows some $\tilde{x} \in conv(X)$ such that $A\tilde{x} = b$; this guarantees that $\rho = 1$, $\theta_B = 0$, $x = z = 0$ is feasible for (33). Dually, (26) is always bounded above since $\bar{f}_{\mathcal{B}} \le f_{\tilde{x}} = c\tilde{x}$.
- Using stabilizing terms that are *not* smooth at zero is possible provided that $\mathcal{D}_t \to 0$ (pointwise) as the algorithm proceeds; basically, this turns S^2DW into a penalty approach to (1), as $\mathcal{D}_t^* \to I_{\{0\}}$. In this case, it is also possible to limit changes of the center $\bar{\pi}$, up to *never* changing it.

– Constraints on the handling of t can be significantly relaxed, up to allowing it to converge to zero, provided that $\mathcal{D}_t$ is "regular enough" as a function of t; for instance, $\mathcal{D}_t = (1/t)D$ for some D satisfying (i) and (ii) (see the Appendix for details).
– The descent test (31) can be weakened by using $v(29)$ in place of $f_{\mathcal{B}}(\tilde{\pi}) - f(\bar{\pi})$, making it easier to declare a SS.

The reader interested in these details is referred to [19]. Although not the most general, the above scheme is already flexible enough to accommodate many algorithmic variants that have proven to be useful in some applications:

– (iii) allows staying at $\bar{\pi}$ even if a "sizable" ascent could be obtained (only provided that this does not happen infinitely many times); this allows for alternative actions to be taken in response to a "good" step, e.g., increasing $t \Rightarrow$ "flattening" $\mathcal{D}_t$.
– (iv) allows solving (2) at other points than $\tilde{\pi}$, or not adding the resulting items to $\mathcal{B}$, at some iterations; this is useful for instance to accommodate a further search on the dual space "around" $\tilde{\pi}$, such as a linear or curved search.
– (v) allows a great deal of flexibility in managing the stabilizing term; the only requirement is that it "never becomes too steep" (but imposing a fixed lower bound on t may not be necessary; see the Appendix) and that changes must be inhibited at some point during very long sequences of NSs. This allows for many different actual strategies for updating t, which are known to be important in practice.

4.2 Choice of the stabilizing terms

The S^{2}DW algorithm does not depend on the choice of the stabilizing term $\mathcal{D}_t$, provided that the above weak conditions are satisfied. Indeed, (29) shows that the choice of $\mathcal{D}_t$ only impacts the $\mathcal{D}_t^*(-z)$ term in the objective function, allowing for many different stabilizing functions to be tested at relatively low cost in the same environment. A number of alternatives have been proposed in the literature for the stabilizing function $\mathcal{D}_t$ or, equivalently, for the primal penalty term $\mathcal{D}_t^*$. In all cases, $\mathcal{D}_t$ is separable on the dual, and therefore $\mathcal{D}_t^*$ is such on the primal, that is,

$$\mathcal{D}_t(\pi) = \sum_{i=1}^{m} \Psi_t(\pi_i) \qquad \mathcal{D}_t^*(z) = \sum_{i=1}^{m} \Psi_t^*(z_i)$$

where $\Psi_t : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is a family of functions. We experimented with two simple versions: $\Psi_t = I_{[-t,t]}$, which establishes a trust region of radius t around the current point ("BoxStep"), and $\Psi_t = \frac{1}{2t}(\cdot)^2$ ("proximal bundle"). From the dual viewpoint, these correspond respectively to the linear penalty $\Psi_t^* = t|\cdot|$ and the quadratic penalty $\Psi_t^* = \frac{1}{2}t(\cdot)^2$. Actually, the treatment could be easily extended to the case when the stabilizing term depends on multiple parameters instead of just one, such as in [6], but we kept one single parameter t for simplicity.

5 Computational experiments

All experiments were performed on a single CPU of a computer with 16 Intel Xeon X7350 CPUs running at 2.93 GHz and 64 Gb of RAM, running Linux Suse 11.1. All the LPs have been solved with CPLEX 11.1.

5.1 Residual capacity inequalities

We start by remarking that an entirely different approach for improving the lower bound $v(\bar{I})$ attains the same result as the decomposition ones. This is based on devising valid inequalities which cut out some of the fractional solutions of $\bar{I}$; in particular, the *residual capacity* inequalities [2,28] consider separately any single arc $(i, j) \in E$. Thus, in the derivation that follows, for notational simplicity, we drop the arc index. Let $a^h = d^h/u$ and, for any subset $P \subseteq H$ of the commodities, define $d^P = \sum_{h \in P} d^h$, $a^P = d^P/u$, $q^P = \lceil a^P \rceil$, and $r^P = a^P - \lfloor a^P \rfloor$; the corresponding residual capacity inequalities can then be written as

$$\sum_{h \in P} a^h (1 - w^h) \geq r^P (q^P - y). \tag{34}$$

These inequalities are valid and easy to separate for any given $[\bar{w}, \bar{y}]$, where $\bar{y}$ is fractional: one simply defines $P = \{h \in H \ : \ \bar{w}^h > \bar{y} - \lfloor \bar{y} \rfloor\}$ and checks if

$$\lfloor \bar{y} \rfloor < a^P < \lceil \bar{y} \rceil \quad \text{and} \quad \sum_{h \in P} a^h \left(1 - \bar{w}^h - \lceil \bar{y} \rceil + \bar{y}\right) + \lfloor \bar{y} \rfloor (\lceil \bar{y} \rceil - \bar{y}) < 0.$$

If so, then (34) corresponding to this P is violated, otherwise there are no violated residual capacity inequalities. Adding all the (exponentially many) inequalities (34) to I produces a (much larger) model $I+$ whose continuous relaxation $\bar{I}+$ is "equivalent" to the DW (and therefore to the SDW and S^{2}DW) approach(es) in the sense that $v(\bar{I}+) = v(\bar{B}+) = v$ (1) (see [21] for details). This approach can be easily implemented using the current, efficient, off-the-shelf MILP solvers, and it has been proven to be competitive with (the non-stabilized) SDW on some classes of instances [21].

5.2 Summary of algorithmic approaches

A remarkably large number of different algorithmic approaches exist for computing the same lower bound:

1. the DW approach applied to the original model I, either in the aggregated or in the disaggregated form;
2. the *Stabilized* DW (also known as bundle method) approach applied to the original model I, again in the two possible aggregated or disaggregated forms;
3. the *Structured* DW approach applied to model $B+$;

4. the *Stabilized Structured* DW approach applied to model $B+$;
5. the completely different cutting-plane algorithm in the primal space, using residual capacity inequalities, applied to solve $\bar{I}+$.

All these approaches provide the same lower bound, and all, except the last one, obtain that by, in fact, maximizing the *very same* dual function defined by (9–12). Yet, while being (almost) all based on the same idea, they have surprisingly little in common, apart from the fact that all of them consider some "very large" reformulation of the original problem. The underlying reformulations have either very many columns and a few rows, or very many rows and relatively few columns, or an "intermediate" (albeit still very large) number of both rows and columns. The problems to be solved at each iteration may be either LPs or QPs, and be either very specially structured (so as to allow specialized approaches [17]) or rather unstructured. Implementing them may require little more than access to a general-purpose LP solver, such as in the case of 5 where the dynamic generation of rows can be handled by the standard `callback` routines that are provided for the purpose. Other cases, such as 2, can be solved by general-purpose bundle codes such as that already used with success in several other applications [10,22,23]. Yet, other cases, such as 3 and 4, require development of entirely ad-hoc approaches. The algorithms may either be basically "fire and forget," or require nontrivial setting of algorithmic parameters. In particular, for stabilized approaches, the initial choice of $\bar{\pi}$ and the management of t can have a significant impact on performances. Regarding the first, $\bar{\pi} = 0$ is the standard choice at the first iteration; however, for the MCND, better options exist. For instance, one may solve the $|H|$ separate shortest path problems corresponding to constraints (5) and (7), with $d^h c^h_{ij} + f_{ij}/u_{ij}$ as flow costs; this corresponds to solving the LP relaxation $\bar{I}$, and produces *node potentials* $\bar{\pi}^h_i$ which can be used as starting point. Alternatively, or in addition, a few iterations of a subgradient-like approach can be performed to quickly get a better dual estimate. This has been tested, because for some applications the choice of the starting point has been shown to have substantial effects on the impact of stabilization on the performances of the algorithms. Indeed, for "simple" problems, a good warm-start can make stabilization almost useless [6], and since the focus of this paper is on stabilization, it is relevant to computationally test to what extent this is (or not) the case.

For non-stabilized approaches, the choice of the initial point is known [6] to have little impact on the performances. This is largely true also for the stabilized DW solved by the standard bundle code, which typically requires many steps anyway, and can usually recover a solution at least as good as the one provided by the initialization in a small fraction of these. Furthermore, several sophisticated approaches for the critical on-line tuning of t [18, I.5] have been devised which helps keep t "large," thereby allowing "long steps" and fast convergence, whenever $\bar{\pi}$ is "far" from the optimum, while t is reduced to enhance the locality properties as the optimum is approached. By contrast, S^2DW terminates in far fewer iterations, each one being significantly more costly due to the larger master problem; therefore, the choice of the initial $\bar{\pi}$ can have a larger impact on performances. Hence, apart from the shortest path warm-start (which can be used by default, since it is very inexpensive), we tested a two-level warm-start where the shortest-path-produced $\bar{\pi}$ is further enhanced by a few iterations

of a subgradient method. Furthermore, the t-strategies are not entirely straightforward to implement within the S^{2}DW setting, especially when the stabilizing term is not the standard quadratic one; hence, for S^{2}DW, we kept t fixed throughout to a hand-tuned value depending on the choice of the initial point. The tuning was done among a few choices, and kept fixed for all instances in the same class. As expected, the "best" t is typically smaller for the two-level warm-start than for the shortest path warm-start.

All the approaches compute the same lower bound to the optimal value of the MCND, but of course this is mostly relevant as a step towards finding the optimal—or at least a provably good—solution to the original integer problem. It is therefore interesting to gauge "the quality" of the "partial" model generated by each approach at termination. To do that, we heuristically solved each of the corresponding MILP models by running CPLEX for one hour with the *polishing* option. This was done for all models except the ones generated by the DW approaches, which anyway, as we will see shortly, are not competitive with the others (see [30,31] for ways to derive integer solutions from DW methods). Note that, of all partial models obtained by the different approaches, only the MILP formulation derived from $\bar{I}+$ is guaranteed to contain an optimal solution to I; for all the others, some columns needed to represent any optimal integer solution might not be in the model.

5.3 Computational results

The experiments have been performed on 88 randomly generated problem instances, already used in [21], to which the reader is referred for more details. The instances are divided into three classes, "medium," "large" and "huge," with $(|N|, |H|)$ respectively equal to (30, 100), (20, 200), and (30, 400). For each class, 8 instances were generated, each with 4 different values of the parameter $C = |E|(\sum_{h\in H} d^h)/(\sum_{(i,j)\in E} u_{ij})$. When $C = 1$ the average arc capacity equals the total demand and the network is lightly capacitated, while it becomes more tightly capacitated as C increases.

We first ran an initial set of tests on a subset of the instances to get an assessment of the effectiveness of each approach and to tune the algorithmic parameters where necessary. From these tests, we could conclude that:

- As expected, due to its instability, the standard (non-stabilized) aggregated DW method could not reach the relative precision of `1e-6` in reasonable time, dramatically tailing off and effectively stopping to converge while still far from the expected value. Unlike in other applications [25], turning to the disaggregated model did not substantially change the outcome.
- Not even the stabilized aggregated DW approach could reach convergence in reasonable time; while the method was indeed converging, the speed was exceedingly slow. This could be expected in view of the results on a similar problem [10]. However, unlike in the non-stabilized case, the disaggregated variant improved things very substantially, resulting in a workable solution.
- The SDW method was always workable, although rather slow for very large instances. The S^{2}DW worked well, but there was no clear dominance between the quadratic stabilization ($\Psi_t = 1/(2t)(\cdot)^2$) and the linear stabilization ($\Psi_t = I_{[-t,t]}$), thus we had to experiment on both.

We then ran a complete set of tests on the remaining approaches, i.e., the primal cutting-plane approach using residual capacity inequalities to solve model $\bar{I}+$ (denoted by "PCP"), the disaggregated stabilized DW method on the integer model I (denoted by "StabDW"), the non-stabilized SDW method on the binary model $B+$ (denoted by "StructDW"), and three versions of the S^2DW method on the binary model $B+$: with quadratic stabilization (denoted by "S^2DW_2"), with linear stabilization (denoted by "S^2DW_∞"), and with linear stabilization and two-level warm-start using the subgradient method (denoted by "S^2DW_∞–ws^2"). In all approaches, the added columns (and the added rows, for PCP) are never removed, nor aggregated. The results are shown in Tables 1, 2, 3. For each instance, we report the improvement (column "imp"), in percentage, between the "weak" lower bound $v(\bar{I})$ and the "strong" lower bound $\underline{v} = v(\bar{I}+) = v(1) = v(\bar{B}+)$, computed as $\text{imp} = 100 \times (\underline{v} - v(\bar{I}))/v(\bar{I})$. To compare the approaches, we report total CPU times (column "cpu"); we remark that all the approaches spend basically all the time in the master problem, with the solution of the Lagrangian subproblem always taking less than 1 % of the time, and as little as 0.01 % for the largest instances. With the exception of StabDW, we also report the gap in percentage (column "gap") between the upper bound, $\bar{v}$, obtained by performing CPLEX heuristics on the MILP model derived for each approach, and the lower bound: $\text{gap} = 100 \times (\bar{v} - \underline{v})/\underline{v}$. For all approaches, we report the total number of iterations (column "it"); for S^2DW, we also report the number of SSs (column "ss").

For the 32 medium instances (Table 1), the StabDW approach is the fastest for eight instances (the easier ones to solve for all approaches), it is somewhat competitive for six other instances, but loses badly—due to a very large number of iterations—for all the others. The SDW approach is often the best for small values of C, but suffers a significant degradation of performances (up to five-fold) as C grows from 1 to 16, making it less competitive for the largest values of C. The quadratic stabilization in S^2DW does not always translate into fewer iterations than the non-stabilized method, although this does indeed happen for $C = 16$; however, the cost of solving the quadratic master problem with CPLEX is very high, especially for some instances, making it very unattractive. We remark here that both the active-set ("simplex") and interior-point ("barrier") algorithms of CPLEX have been tested, with the former proving (as expected) more efficient due to its better reoptimization capabilities; yet, this was not sufficient to achieve good performances. The linear S^2DW is worse than the quadratic one in iteration count, but much better in running time. Furthermore, it is much more stable than the non-stabilized SDW as C grows; while the latter is usually faster for $C \leq 4$, the reverse happens for $C \geq 8$. The effect of the two-stage warm-start is somewhat erratic for these instances, seldom being of any use in terms of CPU time; however, the gaps (where nonzero) are most often substantially reduced. The PCP approach is generally comparable to S^2DW_∞ in performance, often (but not always) being better; yet, in terms of gap, S^2DW_∞ clearly dominates, especially for the most "difficult" instances where the final gaps are larger.

The trends seen for the medium instances are confirmed and amplified in the large ones (Table 2). The (disaggregated) stabilized DW requires more than 100,000 iterations on average, and therefore is typically very slow. The non-stabilized SDW suffers the same sharp degradation of performance as C increases, with almost an order of magnitude difference in one case. A completely opposite trend, barely

Table 1 Results for medium instances

Problem			PCP			StabDW		StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞–ws^2			
$\|A\|$	C	Imp	Cpu	Gap	It	Cpu	It	Cpu	Gap	It	Cpu	Gap	It	ss	Cpu	Gap	It	ss	Cpu	Gap	It	ss
517	1	76.58	9.33	0.14	21	31.14	3556	7.18	0.10	25	79.34	0.09	27	11	10.87	0.09	42	15	11.71	0.09	39	25
	4	68.29	8.55	0.09	20	31.90	3606	6.70	0.10	21	97.08	0.09	31	13	10.81	0.09	39	15	11.10	0.09	42	18
	8	58.46	11.69	0.09	24	30.83	3595	8.08	0.10	21	61.60	0.09	23	12	9.35	0.09	35	12	8.03	0.09	31	13
	16	43.01	11.26	0.43	20	46.07	3742	15.80	0.43	24	96.36	0.43	27	13	12.91	0.43	37	13	8.02	0.42	24	13
517	1	187.00	348.18	5.78	26	4323.41	88144	296.30	6.94	55	16380.00	6.57	51	15	223.22	2.97	66	58	357.38	1.52	91	84
	4	138.22	362.02	6.42	25	3581.13	79390	312.13	7.48	44	17091.70	5.87	47	12	298.34	2.72	70	54	269.85	1.48	69	60
	8	100.08	305.33	6.12	21	4054.19	88807	633.14	6.11	61	22176.20	7.16	37	14	279.88	2.70	64	34	276.91	1.44	65	47
	16	60.49	249.12	6.20	21	3015.71	71651	1138.46	6.45	87	27033.90	6.08	43	18	190.24	2.78	60	21	118.59	1.52	40	18
517	1	56.66	4.80	0.12	21	9.08	1519	3.73	0.03	21	11.68	0.03	28	12	5.30	0.03	35	12	8.34	0.03	52	22
	4	51.68	4.86	0.03	21	8.57	1509	3.97	0.03	21	9.96	0.03	22	12	4.89	0.03	33	12	3.97	0.03	22	10
	8	45.42	5.16	0.03	22	8.43	1488	5.06	0.03	25	9.92	0.03	21	11	4.26	0.03	26	10	5.43	0.03	31	14
	16	35.19	4.74	0.69	18	33.65	3777	8.16	0.69	29	15.04	0.73	27	13	6.68	0.52	36	13	3.83	0.43	19	8
517	1	155.19	140.92	3.95	23	2899.18	69500	188.22	4.70	60	5802.88	4.01	42	13	204.94	2.56	71	57	222.32	1.43	85	71
	4	122.84	194.00	3.87	26	2799.31	65229	147.30	4.15	39	6453.45	4.32	39	15	215.22	2.43	79	40	91.40	1.39	41	36
	8	93.00	151.01	3.96	20	2823.68	66025	354.67	4.31	67	5752.64	4.40	31	12	166.92	2.38	62	25	124.17	1.42	50	21
	16	59.68	115.99	4.72	18	2171.57	56184	551.12	4.94	70	10154.30	5.07	40	14	162.76	2.76	61	20	113.23	1.53	50	19
669	1	74.12	6.27	0.00	14	2.62	655	4.94	0.00	19	40.72	0.00	26	13	8.67	0.00	40	16	13.78	0.00	54	33
	4	66.28	6.98	0.00	15	2.74	657	6.75	0.00	30	50.74	0.00	30	16	7.09	0.00	34	13	8.40	0.00	31	23
	8	57.33	6.90	0.00	15	3.06	723	6.98	0.00	23	37.37	0.00	24	13	7.72	0.00	35	13	7.51	0.00	42	12
	16	43.23	6.05	0.02	13	3.16	803	10.63	0.02	27	33.36	0.02	26	13	9.08	0.02	42	10	8.26	0.02	41	13

Table 1 continued

Problem			PCP			StabDW		StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞–ws^2			
$\|A\|$	C	Imp	Cpu	Gap	It	Cpu	It	Cpu	Gap	It	Cpu	Gap	It	ss	Cpu	Gap	It	ss	Cpu	Gap	It	ss
669	1	114.50	80.33	0.50	26	330.03	11273	36.49	0.46	32	2405.96	0.46	47	15	84.37	0.41	76	48	77.74	0.33	72	66
	4	97.32	78.24	0.46	22	326.88	10951	66.31	0.46	50	1964.43	0.46	45	14	66.86	0.41	74	24	81.00	0.33	73	56
	8	79.62	68.01	0.46	19	322.55	11173	55.29	0.46	33	1974.25	0.46	44	15	49.63	0.41	57	18	39.59	0.33	49	20
	16	56.19	58.16	0.74	19	274.54	9979	164.48	0.81	65	1408.34	0.80	38	17	47.33	0.61	52	16	44.39	0.40	52	22
669	1	55.23	4.59	0.00	16	0.58	246	3.13	0.00	18	8.47	0.00	20	11	5.25	0.00	34	13	6.38	0.00	30	20
	4	50.50	4.34	0.00	16	0.61	246	2.97	0.00	16	6.93	0.00	14	10	5.09	0.00	30	13	5.49	0.00	33	13
	8	44.58	3.66	0.00	13	0.61	265	4.08	0.00	22	7.58	0.00	17	11	5.03	0.00	31	12	5.50	0.00	36	12
	16	34.97	4.41	0.16	15	3.13	745	6.30	0.16	29	10.86	0.16	21	12	5.24	0.16	31	12	4.16	0.17	27	8
669	1	88.97	21.42	0.51	24	27.17	2714	13.97	0.25	33	416.23	0.25	45	19	29.65	0.25	84	28	31.46	0.16	65	58
	4	78.13	19.65	0.25	22	28.88	2768	13.60	0.25	31	293.43	0.25	40	13	20.22	0.25	53	20	22.85	0.16	55	38
	8	66.21	22.23	0.25	21	30.89	2799	18.33	0.25	30	234.38	0.25	39	12	15.64	0.25	40	15	12.17	0.16	37	15
	16	48.93	18.06	0.32	17	244.94	9618	33.94	0.32	33	222.54	0.32	37	16	14.25	0.32	35	12	13.62	0.25	38	15
Avg.		75.25	73.01	1.45	20	858.45	21042	129.01	1.56	36	3760.68	1.51	33	13	68.37	0.80	49	22	63.02	0.48	46	28

Table 2 Results for large instances

Problem			PCP			StabDW		StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞–ws^2			
$\|A\|$	C	Imp	Cpu	Gap	It	Cpu	It	Cpu	Gap	It	Cpu	Gap	It	ss	Cpu	Gap	It	ss	Cpu	Gap	It	ss
229	1	162.59	6842	14.94	70	8525	117458	279	7.29	43	5446	8.55	24	11	449	3.10	71	64	359	1.41	86	68
	4	117.19	6661	13.55	71	8796	115025	441	9.02	58	10997	8.02	36	17	332	3.09	55	42	466	1.38	73	69
	8	83.49	4850	9.92	63	9098	119791	978	8.51	57	19742	7.29	26	12	424	3.02	55	36	381	1.36	57	40
	16	48.68	1651	8.29	50	5785	83284	1813	7.15	75	38765	7.48	37	15	372	2.37	60	22	284	1.10	52	28
229	1	205.67	49081	28.16	109	11748	154821	525	10.50	44	17660	12.11	32	17	860	4.16	76	73	907	1.32	129	119
	4	131.24	30899	25.40	91	9132	131674	807	13.58	45	27326	10.20	29	15	1091	2.79	89	87	1460	1.23	126	118
	8	84.61	16502	21.80	87	12682	162766	1593	10.17	44	83226	10.12	40	17	1027	3.03	78	61	1237	1.20	99	77
	16	42.78	2090	5.59	54	6541	97952	2630	9.20	73	108453	9.21	54	16	399	2.12	65	31	804	1.02	114	73
229	1	147.99	3255	11.58	58	7100	106198	231	7.94	41	4030	7.57	24	12	309	2.78	62	50	242	1.25	67	56
	4	110.93	3198	12.38	57	8114	118176	219	7.82	34	6684	6.88	31	16	300	3.00	57	38	297	1.21	65	48
	8	80.92	2917	7.32	56	8844	110752	510	7.77	41	13273	6.70	37	13	242	2.69	41	30	281	1.28	54	39
	16	49.19	1300	5.87	49	7748	106260	1388	7.09	62	33357	7.81	39	19	249	2.15	42	17	251	1.06	53	22
229	1	185.17	18326	20.53	86	9261	132963	380	7.44	39	10173	****	29	14	557	2.61	80	71	592	1.30	101	95
	4	125.39	15537	18.81	80	11791	147879	612	9.36	49	12638	10.33	25	15	755	2.87	80	68	930	1.22	98	95
	8	85.31	9500	13.08	74	10702	146727	1647	8.87	68	32405	10.61	30	14	468	2.75	50	43	761	1.33	83	66
	16	46.09	1900	7.19	52	7268	107197	3167	7.99	108	69562	8.32	47	17	476	2.22	67	30	357	1.10	53	39
287	1	152.25	3186	12.01	52	8938	122231	269	8.77	42	6791	****	30	13	587	3.51	79	66	378	1.55	74	69
	4	118.18	3119	11.77	56	8622	116606	275	8.56	38	6187	8.05	28	12	272	3.59	47	35	372	1.63	66	62
	8	88.39	2887	11.21	51	6707	99283	668	8.14	61	14935	8.69	33	14	463	3.61	57	40	328	1.52	49	40
	16	55.39	1233	7.11	35	7327	99542	1795	7.80	92	46638	6.38	41	16	345	3.01	56	19	230	1.43	40	21

Table 2 continued

Problem			PCP			StabDW		StructDW			S^2DW_2				S^2DW_∞				S^2DW_∞–ws^2			
$\|A\|$	C	Imp	Cpu	Gap	It	Cpu	It	Cpu	Gap	It	Cpu	Gap	It	ss	Cpu	Gap	It	ss	Cpu	Gap	It	ss
287	1	198.87	14559	27.86	66	8815	120614	598	12.54	53	20949	16.31	39	15	1019	3.92	98	93	1327	1.65	149	143
	4	136.97	11934	22.52	62	8426	112308	603	15.07	37	18258	13.78	27	15	1001	3.72	90	79	891	1.60	98	94
	8	92.94	9656	15.28	64	10098	130536	1221	10.38	41	51703	11.81	29	14	909	3.68	73	50	1040	1.63	102	96
	16	53.45	3579	11.60	54	6801	98972	3515	9.06	99	132097	10.11	54	17	513	2.93	59	25	555	1.26	62	45
287	1	144.47	2082	11.33	47	7398	104285	241	7.30	45	5892	7.09	36	13	348	3.46	60	48	404	1.63	81	73
	4	114.14	2141	8.34	49	7234	107048	207	9.13	34	7166	7.35	35	14	240	3.55	44	36	328	1.65	66	59
	8	86.69	1723	10.36	42	6125	90230	445	9.20	50	11131	****	38	16	340	3.22	52	31	277	1.51	47	39
	16	55.10	1049	7.17	34	6154	91064	1180	6.96	73	29143	6.88	40	14	297	3.20	50	17	165	1.48	35	18
287	1	190.82	13162	18.61	74	11493	152106	577	13.15	54	15691	12.53	39	13	1028	3.76	97	91	749	1.67	107	96
	4	134.41	12015	25.90	72	12067	151753	829	13.04	65	21437	****	40	13	686	3.88	71	60	919	1.74	101	96
	8	92.71	8032	15.56	61	9911	130539	1478	12.58	60	88613	14.62	45	17	759	3.64	66	50	680	1.59	66	55
	16	54.21	2533	8.82	44	6945	103842	1936	8.59	59	133781	8.12	58	17	436	2.85	51	27	455	1.44	52	40
Avg.		108.63	8356	14.06	62	8642	118434	1033	9.37	56	34505	20.72	36	15	549	3.13	65	48	585	1.40	78	66

Table 3 Results for huge instances

Problem			StabDW		StructDW			S^2DW_∞				S^2DW_∞–ws^2			
$\|A\|$	C	Imp	Cpu	It	Cpu	Gap	It	Cpu	Gap	It	ss	Cpu	Gap	It	ss
519	1	100.83	87695	248746	9839	9.96	157	2473	2.23	76	55	1857	2.31	53	38
	4	92.54	88031	247864	9087	11.25	140	2140	2.33	68	54	2487	2.36	66	44
	8	82.16	88918	258266	11613	8.47	143	2338	2.45	66	45	1813	2.30	52	30
	16	65.53	85384	238945	38617	10.26	242	3403	2.66	77	39	2570	2.26	58	23
519	1	140.14	95890	267645	21405	18.41	115	8741	3.14	107	94	7576	3.39	80	67
	4	121.46	103067	266754	25651	18.35	121	11101	3.17	118	94	7722	3.07	75	59
	8	101.19	95873	273728	47618	20.57	160	10020	3.00	103	74	8904	3.03	75	49
	16	73.67	92737	240947	60050	20.04	89	8245	3.00	72	52	8004	3.29	63	24
519	1	88.94	61543	187836	5153	5.75	128	1482	2.19	63	52	1139	1.91	49	32
	4	82.77	60911	188733	8207	6.25	205	1320	1.87	60	41	1341	1.77	53	32
	8	74.84	62377	187785	8490	4.87	163	1649	1.90	71	40	1262	1.84	47	30
	16	61.42	75193	220169	17422	6.35	202	1636	1.99	59	33	1522	2.11	55	19
519	1	125.07	93065	258054	22246	14.90	165	4811	3.31	87	76	4668	3.06	66	55
	4	111.02	90573	250854	17976	18.22	131	4324	2.57	77	64	4373	3.19	66	45
	8	94.82	93418	256884	30460	18.18	159	5224	3.14	85	60	4209	2.86	57	36
	16	71.31	93567	265663	74447	16.50	176	5532	3.14	67	46	5191	3.02	64	23
668	1	126.02	98789	246702	23771	11.89	149	9215	2.96	97	78	6815	3.01	69	56
	4	115.29	99014	247620	28567	10.97	176	6766	2.99	79	63	6506	3.07	69	45
	8	102.03	104481	258636	27871	12.07	130	7560	2.67	87	56	5765	2.78	61	37
	16	80.96	103011	278905	58363	13.95	156	8626	3.14	83	45	3764	2.95	41	18
668	1	111.16	92855	243448	13119	8.69	120	4316	2.53	77	63	4301	2.52	71	44
	4	103.21	94363	243804	12586	8.27	114	4117	2.36	79	51	2695	2.55	46	34
	8	93.15	88587	250759	20560	9.20	164	6314	2.31	102	64	3203	2.54	59	21
	16	76.10	101200	270216	38503	8.75	184	6360	2.30	95	44	3441	2.50	55	19
Avg.		95.65	89606	246055	26317	12.17	154	5321	2.64	81	58	4214	2.65	60	37

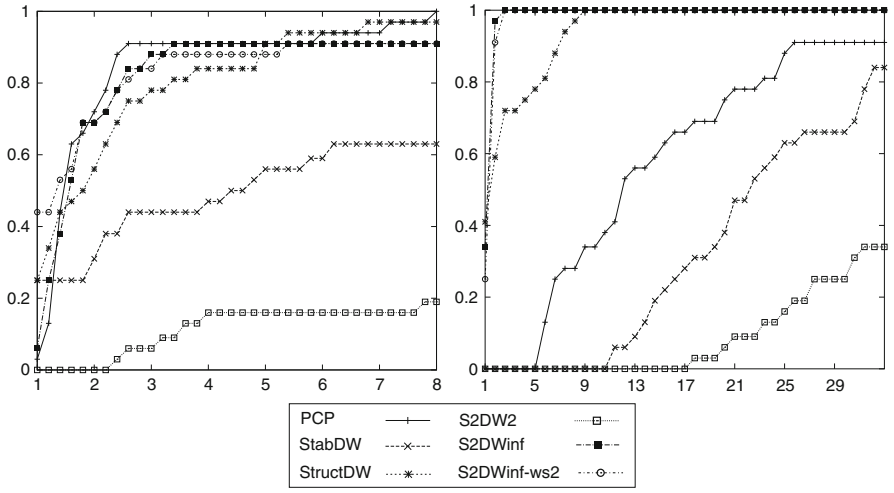

Fig. 3 CPU performance profiles for medium (*left*) and large (*right*) instances

discernible already for the "hard" medium instances, reveals itself for PCP: the approach is significantly faster for large C than for small C, with the ratio between $C = 1$ and $C = 16$ reaching almost 25 in one case. Thus, while StructDW is faster than PCP for $C = 1$, typically by more than an order of magnitude, PCP wins in a few cases for $C = 16$, although at most by 50 %. A similar trend is observed for gaps: StructDW is better (often considerably so) for $C \leq 8$, while in a few cases PCP attains a better gap for $C = 16$. The quadratic S^2DW now outperforms all other approaches in terms of iteration count; however, the cost of solving the quadratic master problem with CPLEX attains intolerable levels. Furthermore, the gaps are not quite as good as those of the linearly stabilized versions; in particular, in four cases (marked with "****" in the Table), no feasible solution at all was found. Here, the linear S^2DW outperforms the competition: only the non-stabilized SDW is faster for small C, but S^2DW is much less affected by the growth of C (actually, most often than not it behaves better for large C than for small ones), being a factor of two faster on average. Furthermore, the gap is substantially smaller than for both StructDW and PCP, irrespective of C. Again, the two-level warm-start has an erratic effect on running times, resulting in a very close average, but halves the gap when compared to the already surprisingly good result of the standard warm-start; this results in an average gap of 1.4 %, which is a full order of magnitude less than the 14 % gap obtained by PCP.

For huge instances (Table 3), we did not compute results for PCP and S^2DW_2, as it is clear from the previous data that they have no hope to be competitive (for PCP, this had already been shown in [21] when comparing it to StructDW alone). The stabilized DW requires around 250,000 iterations to converge, ending up being the slowest in all cases. The non-stabilized SDW suffers from the same dramatic performance decline as C grows, making even the stabilized DW on the original cutting-plane model competitive for $C = 16$. However, S^2DW is much more efficient in time, up to over one order of magnitude, and still delivers much smaller gaps. The two-level warm-start does not have the same uniform effect on gaps as for the medium and large instances, ending

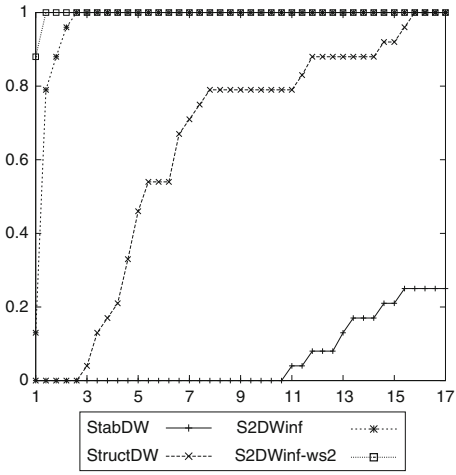

Fig. 4 CPU performance profiles for huge instances

up with very close results; however, the effect on running times is more noticeable, with a reduction between 20 and 50 % being the most common outcome.

The results in the previous tables are summarized in the performance profiles (over CPU times) of the different algorithms shown in Fig. 3 for medium and large instances and in Fig. 4 for huge instances.

These results confirm that using "even more disaggregated" models is typically very beneficial: like moving from aggregated to disaggregated cutting-plane models [3,9,25], the larger model of the SDW approach provides a much faster rate of convergence. For loosely capacitated medium-to-large-scale instances, not stabilizing may be the best choice; this has been reported before for column generation [6]. However, for more tightly capacitated instances, especially as the size grows, stabilization becomes instrumental, provided that care is exercised in choosing the right stabilizing term. The effect on the gap is also noteworthy, since it suggests that stabilization seems capable of helping the SDW approach to select "just the right parts" of the huge model, not only in terms of the solution of the LP relaxation, but also in terms of the integer solution. Exactly why this capability does not appear to be shared by the quadratically-stabilized version, that is otherwise clearly superior in terms of convergence speed, is not clear to us at this point in time, although one may speculate that the linear stabilization "is a lesser change" to the original linear objective function of the master problem. Thus, stabilization not only improves lower bound computation times, but it also appears—at least in this application—useful for constructing actual solutions of the original MILP. Since the non-stabilized SDW approach is a special case of the stabilized one with an "infinitely weak" stabilization term, it is arguably convenient to approach any potential new application with a "stabilize first, ask questions later" strategy.

6 Conclusions and future research

We have analyzed and implemented an extension of the Dantzig–Wolfe decomposition method which exploits the structure of the pricing problem, i.e., the existence of

a *reformulation* of the latter amenable to a column generation procedure. The SDW approach has similar convergence properties to the original DW approach; furthermore, the subproblem that has to be solved in the two algorithms is the same—only the master problem changes—and therefore only limited modifications to existing DW approaches are required to implement SDW methods. The SDW method can be *stabilized* exactly like the original DW approach, leading to the stabilized structured Dantzig–Wolfe method (S^{2}DW); the convergence theory of [19] can be extended to the new approach, providing a reasonably complete picture about what stabilizing terms can be used, how the proximal parameter t and the "bundle" $\mathcal{B}$ can be handled, and so on. We have tested the S^{2}DW approach on the multicommodity capacitated network design problem, obtaining quite encouraging results against a large set of possible alternative approaches providing the same (strong) lower bound.

As far as future developments go, it will be interesting to test more applications of the S^{2}DW approach. For instance, the well-known Gilmore-Gomory formulation of the cutting stock problem [8] is usually solved by DW approaches; there, X is the set of all valid *cutting patterns*, that is, all feasible solutions to an integer knapsack problem. Owing to the well-known reformulation of integer knapsack problems in terms of longest path problems on a directed acyclic network, one can devise the *arc-flow model* of the cutting stock problem [5], that provides the same lower bound at the cost of a pseudo-polynomial number of variables (one for each arc in the graph representing the knapsack) and constraints (one for each node in the same graph). The arc-flow model provides the alternative reformulation of Assumption 1, and each feasible cutting pattern is a path in the directed graph underlying the arc-flow model; thus, one can easily devise a restricted formulation $X_{\mathcal{B}}$ corresponding to a (small) sub-graph of the (large) full directed graph. It will be interesting to verify if the new approach is competitive with existing solution methods, in this application or others. The implementation we tested is also rather naive in terms of the handling of t (fixed) and of $\mathcal{B}$ (no removals, no aggregation); finding appropriate rules for these important aspects has the potential to further substantially improve the computational behavior of the approach.

Acknowledgments We are grateful to the anonymous referees for their valuable comments which helped us to significantly improve the contents of the paper, and to K. Kiwiel for pointing out a flaw in the analysis of [19] (see the Appendix). We are grateful to Serge Bisaillon for his help with implementing and testing the algorithms. We also gratefully acknowledge financial support for this project provided by NSERC (Canada) and by the GNAMPA section of INDAM (Italy).

Appendix: Proof of the convergence results

We now rapidly sketch convergence results for the S^{2}DW method, focusing only on certain aspects where the theory of [19] cannot be directly used due to the above mentioned somewhat weaker assumptions on the update of the model. The standing hypotheses here are (i)—(vi), a *monotone* and *safe* β-strategy, and, at least initially, that X is compact. A basic quantity in the analysis is

$$\Delta f = f_{\mathcal{B}}(\tilde{\pi}) - f(\tilde{\pi}) \geq 0,$$

i.e., the "approximation error" of the model $f_{\mathcal{B}}$ with respect to the true function f in the tentative point $\tilde{\pi}$. It can be shown that if $\Delta f = 0$, then $\tilde{\pi}$ and $(\tilde{x}, \tilde{z})$ are the optimal solutions to the "exact" stabilized problems

$$\begin{array}{l} \max_{\pi} \left\{ f(\pi) - \mathcal{D}_t(\pi - \bar{\pi}) \right\} \\ \min_{z,x} \left\{ cx - \bar{\pi} z + \mathcal{D}_t^*(-z) \ : \ z = Ax - b, \ x \in conv(X) \right\} \end{array} \tag{35}$$

(with $f_{\mathcal{B}} = f$, $X_{\mathcal{B}} = conv(X)$), respectively [19, Lemma 2.2]. This means that $\tilde{\pi}$ is the best possible tentative point, in terms of improvement of the function value, that we can ever obtain unless we change either t or $\bar{\pi}$; in fact, it is immediate to realize that if $\Delta f = 0$, then the "sufficient ascent" condition (31) surely holds. The "inherent finiteness" of our dual function f allows us to prove that this has to happen, eventually, provided that $\mathcal{B}$ is not "treated too badly".

Lemma 2 *Assume that an* asymptotically blocked *β-strategy is employed, i.e., for any sequence of NSs, at length removals from the bundle $\mathcal{B}$ are inhibited; then, after finitely many NSs, either a SS is performed, or the algorithm stops.*

Proof By contradiction, assume that there exists a sequence of infinitely many consecutive NSs (i.e., the algorithm never stops and no SS is done). If an asymptotically blocked β-strategy is employed, during an infinite sequence of consecutive NSs, one has that (31) is (at length, see iii) never satisfied, and this clearly implies $f(\tilde{\pi}) < f_{\mathcal{B}}(\tilde{\pi})$. But, at length, t is fixed by (v), removals from $\mathcal{B}$ are inhibited, no aggregated pieces can be created because the β-strategy is *safe*, and at least one item is added to $\mathcal{B}$ at every iteration by (iv); thus, $\mathcal{B}$ grows infinitely large, contradicting finiteness in Assumption 1. □

Under the stronger version of Assumption 3 where $\bar{x} \in X_{\mathcal{B}'}$, this result could be strengthened to allow reducing the size of $\mathcal{B}$ down to any predetermined value. This first requires the further assumption that $\mathcal{D}_t$ is *strictly convex* (equivalently, $\mathcal{D}_t^*$ is *differentiable*), that is satisfied e.g. by the classic $\mathcal{D}_t = \frac{1}{2t}\|\cdot\|_2^2$ but not by other useful stabilizing terms (cf. e.g. [6]). Then, under the mere monotone β-strategy one can use [19, Lemma 5.6] to prove that the sequence $\{\tilde{z}_i\}$ is bounded, and then [19, Theorem 5.7] to prove that the optimal value v (29) of the stabilized primal master problem is actually *strictly decreasing*; since $\bar{\pi}$ and (at length) t are fixed, this means that no two iterations can have the same $\mathcal{B}$, but the total number of possible different bundles $\mathcal{B}$ is finite. A weaker version (that is sufficient in practice) showing that v (29) $\to 0$ even if infinitely many aggregations are performed is also possible: one could then resort to employing the "poorman cutting-plane model" $f_{\tilde{x}}$ (cf. Sect. 4.1) at all steps, which basically makes the algorithm a subgradient-type approach with deflection [4,13] and results in much slower convergence [10,23]. Thus, this kind of development seems to be of little interest in our case. Instead, the *strictly monotone* β-strategy [19, Definition 4.8], which simply requires suspending removals from $\mathcal{B}$ until v(29) *strictly decreases*, is likely to be an effective way to reduce the size of $\mathcal{B}$ while ensuring the asymptotically blocked property.

Theorem 5 *Under the assumptions of Lemma 2, the sequence $\{f(\bar{\pi}_i)\}$ converges to the optimal value of (3) (possibly $+\infty$). If (3) is bounded above, then a subsequence*

of $\{\tilde{x}_i\}$ *converges to an optimal solution of* (1)*. If, furthermore,* $m = 1$ *then the* S^2DW *algorithm finitely terminates.*

Proof The standing assumption of [19, Sect. 6], i.e., that either the algorithm finitely stops with an optimal solution or infinitely many SSs are performed, is guaranteed by Lemma 2. The fundamental observation is that both f and $f_{\mathcal{B}}$, being value functions of linear programs, are polyhedral. Then, the first statement is [19, Theorem 6.4] (being a polyhedral function, f is $*$-compact). The second statement comes from [19, Theorem 6.2], which proves that the sequence $\{\tilde{z}_i\}$ converges to 0; then, compactness of X implies that a convergent subsequence exists. The third statement is [19, Theorem 6.6]. Note that the latter uses [19, Lemma 6.5], which apparently requires that $f_{\mathcal{B}}$ be the cutting-plane model. However, this is not actually the case: the required property is that $f_{\mathcal{B}}$ be a polyhedral function, and that there exists a "large enough" $\mathcal{B}$ such that $f_{\mathcal{B}} = f$, which clearly happens here. □

The assumption that t is bounded away from zero can be relaxed somewhat if $\mathcal{D}_t = (1/t)D$ for some D satisfying (i) and (ii). It may be useful to remark here that, as correctly pointed out by K. Kiwiel in a private communication, there is a flaw in the treatment of this point in [19]. In particular, the alternative to $t_i \geq \underline{t} > 0$ proposed there is $\sum_{i\to\infty} t_i = \infty$ [19, (6.2)]; that assumption does *not* guarantee that the *whole* sequence $\{\tilde{z}_i\}$ converges to 0, but only existence of a converging subsequence. As a consequence, in the hypothesis of Theorem 6.3 one should replace the "asymptotic complementary slackness" condition $\liminf_{i\to\infty} \tilde{z}_i\bar{\pi}_i = 0$ with the stronger condition that $\tilde{z}_{i_k}\bar{\pi}_{i_k} \to 0$ for some subsequence $\{z_{i_k}\} \to 0$ (which do exist under [19, (6.2)]), or more simply that $\liminf_{i\to\infty} \max\{\|\tilde{z}_i\|, \tilde{z}_i\bar{\pi}_i\} = 0$. Nonetheless, proof of [19, Theorem 6.4]—which is of interest here—only requires convergence of a subsequence, and is therefore valid under the weaker assumption [19, (6.2)].

Note that setting $m = 1$ as required by Theorem 5 to attain finite convergence may come at a cost in practice. In fact, this turns the S^2DW method into a "pure proximal point" approach, where the "abstract" stabilized problems (35) have to be solved to optimality before $\bar{\pi}$ can be updated (it is easy to check that with $m = 1$ a SS can only be performed when $\Delta f = 0$). This is most often not the best choice, computationally [6], and for good reasons. The issue is mostly theoretical; in practice, finite convergence is very likely even for $m < 1$. Furthermore, as observed in the comments to [19, Theorem 6.6], the requirement can be significantly weakened; what is really necessary is to ensure that only finitely many consecutive SSs are performed with $\Delta f > 0$. Thus, it is possible to use any $m < 1$, provided that some mechanism (such as setting $m = 1$ after a while) ensures that sooner or later a SS with $\Delta f = 0$ is performed; once this happens (being $m = 1$ or not), the mechanism can be reset and m can be set back to a value smaller than 1.

The extension to the case where X is *not* compact is relatively straightforward. Stabilization (with the appropriate assumptions) considerably helps in ensuring that the master problems attain optimal solutions even if their feasible regions (in particular, $X_{\mathcal{B}}$) are unbounded. Then, as long as $f(\tilde{\pi}) > -\infty$ Assumption 3 is enough to ensure convergence. If $f(\tilde{\pi}) = -\infty$ instead, one has to assume that the solution of (2) produces a feasible solution $\bar{x}$ and an unbounded descent direction v for $conv(X)$ as a "certificate of unboundedness". Now, because clearly $f_{\mathcal{B}}(\tilde{\pi}) > -\infty = f(\tilde{\pi})$,

the algorithm cannot be stopped; furthermore, it cannot be that the half-line $\{x = \bar{x} + \alpha v, \alpha \geq 0\}$ is entirely contained in $X_{\mathcal{B}}$. Then, "some variables must be missing": as in the finite case, it must be easy to update $\mathcal{B}$ and the associated $\Gamma_{\mathcal{B}}$, $\gamma_{\mathcal{B}}$ and $C_{\mathcal{B}}$ to a $\mathcal{B}' \supset \mathcal{B}$ such that there exists a $\mathcal{B}'' \supseteq \mathcal{B}'$ such that v is an unbounded descent direction for $X_{\mathcal{B}''}$. Once this assumption is satisfied, the theoretical analysis hardly changes. One may loose the second point of Theorem 5 (asymptotic convergence of $\{\tilde{x}_i\}$), but this is of no concern here; *finite* convergence basically only relies on the fact that the total number of possible different bundles $\mathcal{B}$ is finite, so we basically only need to ensure that the same triplet $(\mathcal{B}, \bar{\pi}, t)$ is never repeated. The fact that $conv(X)$ ($\Rightarrow$ some variables θ in its definition) may not be bounded has no impact, provided that the "set of pieces out of which the formulation of $conv(X)$ is constructed" is finite.

References

1. Alvelos, F., Valério de Carvalho, J.M.: An extended model and a column generation algorithm for the planar multicommodity flow problem. Networks **50**(1), 3–16 (2007)
2. Atamtürk, A., Rajan, D.: On splittable and unsplittable flow capacitated network design arc-set polyhedra. Math. Program. **92**, 315–333 (2002)
3. Bacaud, L., Lemaréchal, C., Renaud, A., Sagastizábal, C.: Bundle methods in stochastic optimal power management: a disaggregated approach using preconditioners. Comput. Optim. Appl. **20**, 227–244 (2001)
4. Bahiense, L., Maculan, N., Sagastizábal, C.: The volume algorithm revisited: relation with bundle methods. Math. Program. **94**(1), 41–70 (2002)
5. Belov, G., Scheithauer, G., Alves, C., Valério de Carvalho, J.M.: Gomory cuts from a position-indexed formulation of 1D stock cutting. In: Bortfeldt, A., Homberger, J., Kopfer, H., Pankratz, G., Strangmeier, R. (eds.) Intelligent Decision Support: Current Challenges and Approaches, pp. 3–14. Gabler (2008)
6. Ben Amor, H., Desrosiers, J., Frangioni, A.: On the choice of explicit stabilizing terms in column generation. Discret. Appl. Math. **157**(6), 1167–1184 (2009)
7. Ben Amor, H., Desrosiers, J. Valério de Carvalho, J.M.: Dual-optimal inequalities for stabilized column generation. Oper. Res. **54**(3), 454–4634 (2006)
8. Ben Amor, H., Valério de Carvalho, J.M.: Cutting stock problems. In: Desrosiers, J., Desaulniers, G., Solomon, M.M. (eds.) Column Generation, pp. 131–161. Springer, Berlin (2005)
9. Borghetti, A., Frangioni, A., Lacalandra, F., Nucci, C.A.: Lagrangian Heuristics based on disaggregated bundle methods for hydrothermal unit commitment. IEEE Trans. Power Syst. **18**(1), 313–323 (2003)
10. Crainic, T.G., Frangioni, A., Gendron, B.: Bundle-based relaxation methods for multicommodity capacitated fixed charge network design problems. Discret. Appl. Math. **112**, 73–99 (2001)
11. Croxton, K.L., Gendron, B., Magnanti, T.L.: A comparison of mixed-integer programming models for non-convex piecewise linear cost minimization problems. Manag. Sci. **49**, 1268–1273 (2003)
12. Croxton, K.L., Gendron, B., Magnanti, T.L.: Variable disaggregation in network flow problems with piecewise linear costs. Oper. Res. **55**, 146–157 (2007)
13. d'Antonio, G., Frangioni, A.: Convergence analysis of deflected conditional approximate subgradient methods. SIAM J. Optim. **20**(1), 357–386 (2009)
14. Dantzig, G.B., Wolfe, P.: The decomposition principle for linear programs. Oper. Res. **8**, 101–111 (1960)
15. Elhallaoui, I., Desaulniers, G., Metrane, A., Soumis, F.: Bi-dynamic constraint aggregation and subproblem reduction. Comput. Oper. Res. **35**(5), 1713–1724 (2008)
16. Ford, L.R., Fulkerson, D.R.: A suggested computation for maximal multicommodity network flows. Manag. Sci. **5**, 79–101 (1958)
17. Frangioni, A.: Solving semidefinite quadratic problems within nonsmooth optimization algorithms. Comput. Oper. Res. **21**, 1099–1118 (1996)
18. Frangioni, A.: Dual-Ascent Methods and Multicommodity Flow Problems. PhD thesis, TD 5/97, Dipartimento di Informatica, Università di Pisa, Pisa (1997)
19. Frangioni, A.: Generalized bundle methods. SIAM J. Optim. **13**(1), 117–156 (2002)

20. Frangioni, A.: About Lagrangian methods in integer optimization. Ann. Oper. Res. **139**, 163–193 (2005)
21. Frangioni, A., Gendron, B.: 0–1 reformulations of the multicommodity capacitated network design problem. Discret. Appl. Math. **157**(6), 1229–1241 (2009)
22. Frangioni, A., Gentile, C., Lacalandra, F.: Solving unit commitment problems with general ramp contraints. Int. J. Electr. Power Energy Syst. **30**, 316–326 (2008)
23. Frangioni, A., Lodi, A., Rinaldi, G.: New approaches for optimizing over the semimetric polytope. Math. Program. **104**(2–3), 375–388 (2005)
24. Hiriart-Urruty, J.-B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms volume 306 of Grundlehren Math. Wiss. Springer, New York (1993)
25. Jones, K.L., Lustig, I.J., Farwolden, J.M., Powell, W.B.: Multicommodity Network flows: the impact of formulation on decomposition. Math. Program. **62**, 95–117 (1993)
26. Kiwiel, K., Lemaréchal, C.: An inexact bundle variant suited to column generation. Math. Program. **118**, 177–206 (2009)
27. Lemaréchal, C.: Lagrangian relaxation. In: Jünger, M., Naddef, D. (eds.) Computational Combinatorial Optimization, pp. 115–160. Springer, Heidelberg (2001)
28. Magnanti, T.L., Mirchandani, P., Vachani, R.: The convex hull of two core capacitated network design problems. Math. Program. **60**, 233–250 (1993)
29. Petersen, B., Jepsen, M.K.: Partial path column generation for the vehicle routing problem with time windows. In: Bigi, G., Frangioni, A., Scutellà, M.G. (eds). Proceedings of the 4th International Network Optimization Conference (INOC2009), pages paper TB4-3 (2009)
30. Vanderbeck, F.: On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Oper. Res. **48**(1), 111–128 (2000)
31. Villeneuve, D., Desrosiers, J., Lübbecke, M.E., Soumis, F.: On compact formulations for integer programs solved by column generation. Ann. Oper. Res. **139**, 375–388 (2005)