Mathematical Programming manuscript No.
(will be inserted by the editor)

Bundle Methods for Sum-Functions with "Easy" Components: Applications to Multicommodity Network Design

Antonio Frangioni · Enrico Gorgone

the date of receipt and acceptance should be inserted later

Abstract We propose a version of the bundle scheme for convex nondifferentiable optimization suitable for the case of a sum-function where some of the components are "easy", that is, they are Lagrangian functions of explicitly known compact convex programs. This corresponds to a *stabilized partial Dantzig-Wolfe decomposition*, where suitably modified representations of the "easy" convex subproblems are inserted in the master problem as an alternative to iteratively inner-approximating them by extreme points, thus providing the algorithm with exact information about a part of the dual objective function. The resulting master problems are potentially larger and less well-structured than the standard ones, ruling out the available specialized techniques and requiring the use of general-purpose solvers for their solution; this strongly favors piecewise-linear stabilizing terms, as opposed to the more usual quadratic ones, which in turn may have an adverse effect on the convergence speed of the algorithm, so that the overall performance may depend on appropriate tuning of all these aspects. Yet, very good computational results are obtained in at least one relevant application: the computation of tight lower bounds for Fixed-Charge Multicommodity Min-Cost Flow problems.

Keywords Nondifferentiable Optimization, Bundle methods, Lagrangian Relaxation, Stabilized Partial Dantzig-Wolfe Decomposition, Multicommodity Network Design

Mathematics Subject Classification (2000) 90C06 · 90C25

1 Introduction

We are concerned with the numerical solution of the problem

$$\inf \left\{ f(x) = \textstyle\sum_{k \in \mathcal{K}} f^k(x) \ : \ x \in X \right\} \quad , \tag{1}$$

Antonio Frangioni
Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 2, 56127 Pisa, Italy, E-mail: frangio@di.unipi.it

Enrico Gorgone
Dipartimento di Elettronica Informatica e Sistemistica, Università della Calabria, 87036 Rende (CS), Italy, E-mail: egorgone@deis.unical.it

where $\mathcal{K}$ is a finite set, each $f^k : \mathbb{R}^n \to \mathbb{R}$ is a finite-valued (hence proper) convex possibly nondifferentiable function, and $X \subseteq \mathbb{R}^n$ is closed convex. To simplify the treatment, we will initially assume $X = \mathbb{R}^n$, with the extension to the constrained case to be discussed later on. Our motivation comes from the case where f is the Lagrangian function of a block-structured problem

$$\sup \left\{ \sum_{k\in\mathcal{K}} c^k u^k \ : \ \sum_{k\in\mathcal{K}} A^k u^k = b \ , \ u^k \in U^k \quad k \in \mathcal{K} \right\} \tag{2}$$

w.r.t. the "complicating" constraints that link together blocks of variables that would otherwise be independent, i.e.,

$$f(x) = xb + \sum_{k\in\mathcal{K}} \left(f^k(x) = \sup \left\{ (c^k - xA^k)u^k \ : \ u^k \in U^k \right\} \right) \tag{3}$$

(with a little abuse of notation we use the same index set $\mathcal{K}$ in both (1) and (3), although (1) should have one extra index to account for the linear term xb in (3)). To save on notation, we will often write cu, $Au = b$ and U respectively for the objective function, linking constraints and feasible region in (2)/(3) when the sum-function structure is not relevant. The linearity of objective function and non-linking constraints in (2) is actually not necessary, and it is assumed only for allowing the derivation to use simpler and more familiar forms of duality; a more general setting using *Fenchel's duality* allows to significantly extend the results, among other things requiring only *some* of the f^k to be Lagragian functions, the interested reader being referred to [27] for details.

Bundle methods are known to be among the most efficient implementable algorithms for convex nondifferentiable optimization. We will refer in particular to the *generalized* ones defined in [21], since that is perhaps the largest class of bundle methods with a unified convergence analysis. However, several other bundle-type algorithms have been proposed (see [35] for a thorough discussion and several recent references like [1,32,33,38,40,41,44,45,53]), and the basic idea of the present paper appears to be easily applicable to most of them as well. For (3), it is known that bundle algorithms also solve a "convexified" version of (2) (see e.g. [19,21,46] and (26) below), which is useful e.g. in combinatorial optimization [22,28] and stochastic programming [18,51].

It is a standard assumption in most nondifferentiable optimization algorithms that f is only known through an oracle ("black box") that, given any x ($\in X$), returns the value $f(x)$ and one subgradient z. For our dual application (3), an element $z \in \partial f(x)$ of the subdifferential of f at x is associated to (and conveniently made available by) a primal optimal solution u of the Lagrangian subproblem for the fixed x (although approximate solution of the subproblem is also an option, cf. e.g. [17,38]), which we assume to be able to efficiently compute. The main idea of bundle methods is to sample the feasible space at a finite set $\bar{X} \subseteq X$ of *tentative points*, collecting for each $x_i \in X$ the function value $f(x_i)$ and the specific $z_i \in \partial f(x_i)$ arbitrarily returned by the oracle (of course, $z_i = \nabla f(x_i)$ if f happens to be differentiable in x_i). The corresponding *bundle* $\mathcal{B} = \{ (x_i, f(x_i), z_i) \ : \ x_i \in \bar{X} \}$ is used to construct a *model* $f_{\mathcal{B}}$ of f; typically, the (aggregated) *cutting-plane model*

$$f_{\mathcal{B}}(x) \ = \ \max \left\{ f(x_i) + z_i(x - x_i) \ : \ (x_i, f(x_i), z_i) \in \mathcal{B} \right\} \ \leq \ f(x) \ . \tag{4}$$

This is then used to construct the next tentative point; its simplest choice—that of Kelley's Cutting Plane method—is just a minimum of $f_{\mathcal{B}}$, but some form of *stabilization* (discussed below) is usually necessary in order to improve the performance. Several different forms of stabilization have been analyzed in the literature [1,21,32,33,39,44, 53], but the focus of the present paper is rather the choice of the model $f_{\mathcal{B}}$.

Indeed, it is well-known that exploiting as much as possible the *structure* of the problem at hand is necessary to obtain efficient convex minimization algorithms. This can take many different forms even in "simple" algorithms like subgradient methods, such as exploiting the geometrical structure of the proximal term (be it the Euclidean norm [48] or a Bregman distance [5]), exploiting the sum-function structure to reduce the iteration cost [8], exploiting the composite structure to perform smoothing [49], exploiting the availability of a self-concordant barrier for the feasible region X [50], and others. Not surprisingly, many of these ideas apply to bundle methods (which some subgradient methods are the "poorman's version" of [4]) as well: see e.g. [39] for the use of entropic proximal terms, [17] for reducing the iteration cost in the sum-function case, [56] for the composite structure, [33] for the log-barrier, and [40] for exploiting the structure of X. A significant fraction of the results concern the development of *specialized models* exploiting specific properties of f. For a sum-function such as (3), for instance, a separate oracle is typically available for each component $k \in \mathcal{K}$ providing $f^k(x_i)$ and $z_i^k \in \partial f^k(x_i)$, so that $f(x_i) = \sum_{k\in\mathcal{K}} f^k(x_i)$ $(+x_i b)$ and $z_i = \sum_{k\in\mathcal{K}} z_i^k$ $(+b)$. Thus, a relatively straightforward idea is to separately keep the $|\mathcal{K}|$ *disaggregated bundles* $\mathcal{B}^k$ to construct $|\mathcal{K}|$ *independent models* $f_{\mathcal{B}}^k$, one for each component, so that $f_{\mathcal{B}} = \sum_{k\in\mathcal{K}} f_{\mathcal{B}}^k$ is a much better model—for the same set $\bar{X}$ of tentative points—of f than (4). This *disaggregated model* improves the convergence speed so much as to amply compensate the increased cost due to the larger size of the corresponding master problems [3, 10, 16, 37, 54]. Other relevant cases are piecewise-conic models of piecewise-conic functions [34, 40, 52], and Newton-type models of smooth components f^k of a sum-function [2, 41, 45].

In this paper, we continue on the same vein targeting a different structure that—somewhat surprisingly—does not seem to have been exploited yet: the case where some of the f^k are Lagrangian functions of a "simple" convex program. This allows to use *exact models* of these components by basically just copying the corresponding constraints in the formulation of the master problem, instead of iteratively approximating them by inner linearization. From the primal viewpoint, this amounts to performing a *stabilized partial Dantzig-Wolfe decomposition* where not all the polyhedra whose Cartesian product forms the feasible region are reformulated by extreme points and iteratively inner approximated. This enlarges the initial size of the master problem, but not necessarily the average size since the exact models do not grow in size with the iterations like classical ones do. Furthermore, it provides the algorithm with a better—indeed, "perfect"—knowledge of the corresponding f^k, thereby possibly improving the convergence speed. On the other hand, this destroys the standard structure of the master problem upon which specialized approaches [20] rely, thereby requiring the use of general-purpose solvers for the master problem solution. This in turn strongly favors the use of piecewise-linear stabilizing terms (cf. (27) below), as opposed to the usual quadratic ones, which unfortunately may have an adverse effect on the convergence speed of the algorithm [6, 11, 24]. However, we show that the new approach leads to a very substantial reduction of the overall running time in at least one relevant application: the computation of tight lower bounds for Fixed-Charge Multicommodity Min-Cost Flow problems (FC-MMCF).

The structure of the paper is as follows. In Section 2 we present the application motivating our development, FC-MMCF, and discuss different possible variants of Dantzig-Wolfe decomposition for the problem highlighting the potential for improvements w.r.t. "naïve" approaches. Then, in Section 3 we first provide an overview of

bundle methods, and then we discuss how "exact" models of "easy" components can be inserted into the master problem. Finally, in Section 4 the computational results obtained with FC-MMCF are presented and discussed, and in Section 5 conclusions are drawn.

Throughout the paper the following standard notation is used. For a set X, $I_X(x) = 0$ if $x \in X$ (and $+\infty$ otherwise) is its *indicator function*, and *co* X is its *convex hull*. For a convex function f, *dom* $f = \{ x : f(x) < \infty \}$ is its *domain*.

2 Motivation: decomposition approaches for FC-MMCF

The Fixed-Charge Multicommodity Min-Cost Flow problem (FC-MMCF) is as follows. Given a directed network $G = (N, A)$, where N is the set of nodes and A is the set of arcs, we must satisfy the demands of a set of commodities K. Each commodity $k \in K$ is characterized by a deficit vector $b^k = [b_i^k]_{i \in N}$ indicating the net amount of flow required by the node: nodes with negative deficit are sources, nodes with positive deficits are sinks, and nodes with zero deficits are transshipment. In many (but not all) applications a commodity is an origin-destination pair (s_k, t_k) with an associated demand $d^k > 0$ that must flow between s_k and t_k, i.e., $b_i^k = -d^k$ if $i = s_k$, $b_i^k = d^k$ if $i = t_k$, and $b_i^k = 0$ otherwise. Each arc (i, j) in the network can only be used if the corresponding *fixed cost* $f_{ij} > 0$ is paid; in this case it has a *mutual capacity* $u_{ij} > 0$. Also, *individual capacities* u_{ij}^k are defined for the maximum amount of flow of commodity k on arc (i, j). These may either come from the real application modeled by the problem, or be introduced to strengthen the model; for instance, in the origin-destination case one may set $u_{ij}^k = d^k$, which is useful if $d^k \ll u_{ij}$. Finally, a *routing cost* c_{ij}^k has to be paid for each unit of commodity k moving through (i, j). The problem consists in satisfying demand requirements and capacity constraints at minimal total cost. Defining *arc flow variables* w_{ij}^k for the amount of commodity k flowing on arc $(i, j) \in A$ and binary *design variables* y_{ij} which define whether or not the arc (i, j) has been paid for, the (weak) *arc flow formulation* of the problem is

$$\min \ \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k w_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \tag{5}$$

$$\sum_{(j,i) \in A} w_{ji}^k - \sum_{(i,j) \in A} w_{ij}^k = b_i^k \qquad i \in N \ , \ k \in K \tag{6}$$

$$\sum_{k \in K} w_{ij}^k \le u_{ij} y_{ij} \qquad (i, j) \in A \tag{7}$$

$$0 \le w_{ij}^k \le u_{ij}^k \qquad (i, j) \in A \ , \ k \in K \tag{8}$$

$$y_{ij} \in \{0, 1\} \qquad (i, j) \in A \tag{9}$$

2.1 Decomposition approaches

FC-MMCF has the form (3), and therefore Lagrangian techniques can be used for computing lower bounds that can then be effectively incorporated into either heuristic [14,30,31] or exact [36,42] approaches. Indeed, the multicommodity flow structure has always been a favorite target for Lagrangian approaches [12,13,23,24,29,37, 45]. Here we consider the Lagrangian relaxation of constraints (7) with multipliers $\alpha = [\, \alpha_{ij} \,]_{(i,j) \in A} \ge 0$, which gives the (concave) Lagrangian function

$$f(\alpha) = \left\{ \begin{array}{c} \min \sum_{k \in K} \sum_{(i,j) \in A} (c_{ij}^k + \alpha_{ij}) w_{ij}^k + \sum_{(i,j) \in A} (f_{ij} - \alpha_{ij} u_{ij}) y_{ij} \\ (6) \ , \ (8) \ , \ (9) \end{array} \right.$$

whose computation is easy due to separability. In fact, the feasible region is $W \times Y$, where in turn $W = \bigotimes_{k \in K} W^k$, with each $W^k \subset \mathbb{R}^{|A|}$ being defined by

$$\sum_{(j,i) \in A} w^k_{ji} - \sum_{(i,j) \in A} w^k_{ij} = b^k_i \quad i \in N \qquad , \qquad 0 \leq w^k_{ij} \leq u^k_{ij} \quad (i,j) \in A \ ,$$

i.e., the classical (single-commodity) *min-cost flow* structure, for which plenty of efficient solution algorithms exist [26,29]. Also, linear optimization on the unitary hypercube Y is very easy; actually, Y itself is the cartesian product of the $|A|$ sets $Y^{ij} = \{0,1\}$, each one concerning the single variable y_{ij}. Thus, computation of $f(\alpha)$ (and its subgradients) is quite easy. The well-known downside [22] is that the lower bound is then the same as that of the continuous relaxation, as all subproblems have the integrality property. It is also well-known [22] that solving the corresponding Lagrangian Dual via Kelley's Cutting Plane method is equivalent to the celebrated *Dantzig-Wolfe decomposition approach*, which is nothing but a huge-scale reformulation of the problem solved by column generation. In particular, denoting by $\mathcal{S}$ the set of (the indices of) all possible extreme points $[w^1_s, \ldots, w^k_s, y_s]$ of $W \times Y$, the *Dantzig-Wolfe reformulation* of (the continuous relaxation of) FC-MMCF is

$$\min \ \sum_{s \in \mathcal{S}} \big(\, f y_s + \sum_{k \in K} c^k w^k_s \, \big) \theta_s \tag{10}$$

$$\sum_{s \in \mathcal{S}} \big(\, \sum_{k \in K} w^k_{ij,s} - u_{ij} y_{ij,s} \, \big) \theta_s \leq 0 \qquad (i,j) \in A \tag{11}$$

$$\sum_{s \in \mathcal{S}} \theta_s = 1 \quad , \quad \theta_s \geq 0 \qquad s \in \mathcal{S} \tag{12}$$

where $c^k x^k_s$ and $f y_s$ denote the scalar product between the corresponding vectors. Since $|\mathcal{S}|$ is huge, (10)–(12) can only be solved by column generation: select a (small) subset $\mathcal{B}$ of $\mathcal{S}$, form the *master problem* with only the columns $s \in \mathcal{B}$, and then use the dual variables of the *linking constraints* (11) as Lagrangian multipliers in $f(\alpha)$ to compute a new useful extreme point, or prove that none exists.

It is well-known [37] that *disaggregating* the master problem, exploiting the separability of the feasible region, can pay off significantly. The idea is simply to individually consider all the constituents $w^1_s, \ldots, w^k_s, y_s$ of the produced vertices and assign a separate variable in the master problem to each. This is particularly revealing in FC-MMCF when commodities are origin-destination pairs, since then the extreme points of each W^k correspond to elements $p \in \mathcal{P}^k$ of the set of all s_k–t_k paths. Hence, defining *path flow variables* f_p for each $p \in \mathcal{P} = \cup_{k \in K} \mathcal{P}^k$ and denoting by $\mathcal{H}$ the set of all (indices of the) $2^{|A|}$ vertices y_h of Y, we obtain that the disaggregated version of (10)–(12) corresponds to the *path flow formulation*

$$\min \ \sum_{p \in \mathcal{P}} c_p f_p + \sum_{h \in \mathcal{H}} (f y_h) \theta_h \tag{13}$$

$$\sum_{p \in \mathcal{P} \, : \, (i,j) \in p} f_p \leq \sum_{h \in \mathcal{H}} \big(u_{ij} y_{ij,h} \big) \theta_h \qquad (i,j) \in A \tag{14}$$

$$\sum_{p \in \mathcal{P}^k} f_p = d^k \qquad k \in K \tag{15}$$

$$f_p \geq 0 \qquad p \in \mathcal{P} \tag{16}$$

$$\sum_{h \in \mathcal{H}} \theta_h = 1 \quad , \quad \theta_h \geq 0 \qquad (i,j) \in A \tag{17}$$

where c_p is the cost of the path p (sum of the costs of its arcs). Even cursory examination of (13)–(17) shows that something weird has happened: the very simple set Y, representable by $|A|$ variables and $2|A|$ constraints, has been expressed by means of $2^{|A|}$ variables. Thus, a "blatantly wrong" (arguably the worst possible one) reformulation has taken place for the Y-part of the problem. This is not necessary: one could have considered instead the *partial Dantzig-Wolfe decomposition*

$$\min \ \sum_{p\in\mathcal{P}} c_p f_p + \sum_{(i,j)\in A} f_{ij} y_{ij} \tag{18}$$

$$\sum_{p\in\mathcal{P}\,:\,(i,j)\in p} f_p \le u_{ij} y_{ij} \qquad (i,j)\in A \tag{19}$$

$$(15)\ ,\ (16) \tag{20}$$

$$y_{ij} \in [0,1] \qquad (i,j)\in A \tag{21}$$

where only W has been reformulated, but not Y; note that in (18)–(21) y_{ij} is a variable, while in (13)–(17) $y_{ij,h}$ is a constant, and the variable is θ_h. Of course, one then generates extreme points (paths) of the W^k components, but not of the Y one; an analogous treatment can be applied to the aggregated master problem (10)–(12).

Remarkably, (18)–(21) can also be obtained by performing a *full disaggregation*, where not only W is decomposed into the $|K|$ distinct W^k, but also Y is decomposed into its $|A|$ components Y^{ij}. It is easy to see that this is basically equivalent to (18)–(21): each Y^{ij} has only two extreme points $y_0^{ij}=0$ and $y_1^{ij}=1$, and assigning them multipliers θ_0^{ij} and θ_1^{ij}, respectively, one is in fact performing the substitution

$$y_{ij} = 0\cdot\theta_0^{ij} + 1\cdot\theta_1^{ij} \quad , \quad \theta_0^{ij}+\theta_1^{ij}=1 \quad , \quad \theta_0^{ij}\ge 0 \quad , \quad \theta_1^{ij}\ge 0$$

which is nothing but a convoluted way to write (21). This does *not* mean, however, that the partial Dantzig-Wolfe is just a special case of, or a minor improvement upon, the standard disaggregated master problem approach. Indeed, consider the slight variant of FC-MMCF where one requires that at most a given number $0<H<|A|$ of arcs are "opened": this corresponds to adding to (5)–(9) the simple extra constraint

$$\sum_{(i,j)\in A} y_{ij} \le H \tag{22}$$

whose immediate effect is to kill separability of the Y set. Thus, in such a variant there can be at most $|K|+1$ subproblems, and full disaggregation is no longer viable. However, doing partial Dantzig-Wolfe for this case is straightforward: just add (22) to (18)–(21). Thus, while being equivalent to full disaggregation for FC-MMCF, the partial Dantzig-Wolfe idea is in general different.

It is intuitive (and it will be confirmed in §4) that the partial Dantzig-Wolfe (18)–(21) is preferable to the standard Dantzig-Wolfe (13)–(17): the master problem contains a more compact and "exact" representation of a part of the problem. The issue is that the Dantzig-Wolfe approach, partial or not, in itself is often not efficient enough due to *instability*. In simple words, this is the phenomenon whereby two subsequent dual iterates are very far from each other, even when the first one happens to be "close" to the optimal solution. It has been repeatedly shown (e.g. [11]) that *stabilizing* the Dantzig-Wolfe approach—and its close relatives [6,24]—can have a significant impact on its performances. In primal terms, stabilization can be performed as easily as adding slack variables to the master problem to allow violation of the linking constraints "at a cost"; since the structure of the linking constraints is completely immaterial to this operation, it is clearly possible to develop a *stabilized partial Dantzig-Wolfe* approach as well. For instance, in (18)–(21) this could look like replacing (19) with

$$\sum_{p\in\mathcal{P}\,:\,(i,j)\in p} f_p - u_{ij} y_{ij} \le s_{ij} \tag{23}$$

where a (linear or nonlinear) cost is paid if $s_{ij}>0$. Properly handling all this requires an understanding of the dual side, where one is basically employing a bundle method for a sum-function where some of the f^k are "easy" and can be dealt with in a specialized way. The next section is devoted to the development of such an approach.

3 Bundle methods with "easy" components

We first provide a quick overview of bundle methods. We remark that appropriate convex analysis tools allow to give a primal interpretation to (1) even when f is any convex function provided by an oracle, rather than the explicit dual function of (2), as well as to easily extend the approach both to *non-polyhedral models* and to a much larger set of possible *stabilizing terms* than the simple ones employed in our derivation. However, we prefer to present the approach in a setting that is familiar to any reader, and in particular to those interested in large-scale linear and combinatorial optimization, thus confining ourselves to using just linear and quadratic duality: the interested reader is referred to [1,6,21,27] for details of the more general approaches.

3.1 A brief overview of bundle methods

Bundle methods are best described starting from their dual viewpoint: as already mentioned, using the unabridged cutting plane model to select the next tentative point as the optimal solution of

$$\inf_x \left\{ f_{\mathcal{B}}(x) \right\} \;=\; \inf_{v,x} \left\{ v \;:\; v \geq f(x_i) + z_i(x - x_i) \quad (x_i, f(x_i), z_i) \in \mathcal{B} \right\} \;,$$

produces a highly inefficient approach (which, however, have at least the compelling feature that the master problem is an LP). This is due to the fact that the sequence of points generated in this way oscillates wildly, even when approaching the minima of f; to "damp" this oscillation, one selects a *current point* $\bar{x}$, usually the best iterate found so far, a *proximity parameter* $t > 0$, and solves instead the *stabilized master problem*

$$\inf_d \left\{ f_{\mathcal{B}}(\bar{x} + d) + \tfrac{1}{2t} || \, d \, ||_2^2 \right\} \tag{24}$$

where a *stabilizing term* is added to the model to ensure that the next iterate $x_+ = \bar{x} + \tilde{d}$, for the optimal solution $\tilde{d}$ to (24), is "not too far" from $\bar{x}$. If the model is "accurate enough" then $\tilde{d}$ will be a descent direction, i.e., $f(x_+) < f(\bar{x})$; provided that the decrease in the function value is "sizable", $\bar{x}$ can be moved to x_+ (a *Serious Step*). Otherwise $\bar{x}$ is left unchanged (a *Null Step*) and a triple $(x_+, f(x_+), z_+ \in \partial f(x_+))$ is added to $\mathcal{B}$ in order to improve $f_{\mathcal{B}}$. In this process t might be simply kept fixed, but online tuning it to suitably reflect the actual "trust region" where $f_{\mathcal{B}}$ is a "good" model of f is known to be very important for the practical efficiency of the approach (see e.g. [43,57]); anyway, simple general rules can be given [21] such that any t-strategy obeying them results in a finite (since f is polyhedral) minimizing sequence for (1).

The primal viewpoint of bundle methods follows from the fact that for any triple $(x_i, f(x_i), z_i) \in \mathcal{B}$ we have available one $u_i \in U$ such that $f(x_i) = (c - x_i A)u_i$ and $z_i = b - Au_i$; this means that we can as well consider $\mathcal{B}$ as a subset of U, since any $u_i \in U$ allows to re-construct all the information that is necessary to write down the corresponding linear term in (4) (a closer inspection readily shows that the dual point x_i is actually not needed, since it only contributes with the constant term $x_i z_i$ that can be easily accounted for). Doing so in (24) ends up with a Quadratic Program whose constraints are indexed over $u_i \in \mathcal{B} \subset U$; it is then a simple exercise in (quadratic) duality to show that the dual of (24) is

$$\sup \left\{ \; \textstyle\sum_{u_i \in \mathcal{B}} (cu_i)\theta_{u_i} + \bar{x}z - \frac{t}{2} || \, z \, ||_2^2 \;:\; A \left(\sum_{u_i \in \mathcal{B}} u_i \theta_{u_i} \right) - b = z \;\;,\;\; \theta \in \Theta \right\} \tag{25}$$

where $\Theta = \{ \theta \geq 0 : \sum_{u_i \in \mathcal{B}} \theta_{u_i} = 1 \}$ is the unitary simplex of appropriate dimension. To simplify the notation, in the following we will use $c_i = cu_i$, $\theta_i = \theta_{u_i}$, and the shorthand "$i \in \mathcal{B}$" for "$u_i \in \mathcal{B}$". Yet, (25) becomes more revealing when rewritten as

$$\sup \left\{ cu + \bar{x}(b - Au) - \tfrac{t}{2} || Au - b ||_2^2 \ : \ u \in co\, \mathcal{B} = U_{\mathcal{B}} \right\} \ . \tag{26}$$

Thus, from the primal viewpoint bundle methods combine an *augmented Lagrangian* scheme for solving (2) with an inner linearization approach where U is substituted by its approximation $U_{\mathcal{B}}$, the convex hull of the solutions of the pricing problem (3) collected so far. The optimal solution of (26) in the u-space, $\tilde{u} = \sum_{i \in \mathcal{B}} u_i \tilde{\theta}_i$ where $\tilde{\theta}$ is its optimal solution of (25) in the θ-space, can be shown, under mild conditions, to converge to a solution of the "close-convexified version" of (2) where U is substituted with $co\, U$. This also immediately illustrates the well-known relationships between (1) and (2): the two are equivalent when U is convex. This is easily extended far beyond the linear case to any concave c and convex U satisfying mild assumptions; if c is not concave and/or U is not convex, f is nonetheless a convex function and (1) is equivalent to (and, therefore, bundle methods solve) a convex relaxation of (2) which entails $co\, U$ if c is linear, but is significantly more complex otherwise [46]. These results immediately extend to inequality constraints $Au \leq b$, yielding sign constraints $x \geq 0$ in (1); conversely, the generic nonlinear case $A(u) \leq b$ is considerably more complex, even in the convex case [46].

An important remark is that there is little special about the Euclidean norm in (24)/(26): one could use as well any convex function (satisfying a few mild assumptions [21]) which allows an explicit dual to be computed. From the primal viewpoint, (26) only requires that the original constraints be slackened, and an appropriate cost be paid for their violation: indeed, one could as well opt for a linear cost instead, such as

$$\begin{array}{l} \sup \sum_{i \in \mathcal{B}} c_i \theta_i - \bar{x}(z^+ - z^-) - t(z^+ + z^-) \\ \quad A \left(\sum_{i \in \mathcal{B}} u_i \theta_i \right) - b = z^+ - z^- \ , \ \ \theta \in \Theta \end{array} \tag{27}$$

(that is, "just Dantzig-Wolfe decomposition with slacks", cf. (23)), to end up with

$$\inf_d \left\{ f_{\mathcal{B}}(\bar{x} + d) \ : \ || d ||_\infty \leq 1/t \right\}$$

as the dual. In general, the arguments in this paper would apply as well to the other stabilizing functions proposed in the literature, such as piecewise-quadratic or exponential (cf. [21] and the references therein), entropic [39], piecewise-linear with more than one piece [6], and/or depending on multiple parameters instead of the single t [3, 6]; this is indeed shown in [27], to which the interested reader is referred for details. However, to simplify the treatment we will restrict ourselves to presenting them for simple form (24)/(26) which allows resorting to quadratic duality, knowing that the arguments will a fortiori apply to (27) with the even simpler linear duality.

As previously remarked, when f is a sum-function it is better to construct independent models for each component; the corresponding *disaggregated* master problem

$$\inf_d \left\{ \sum_{k \in \mathcal{K}} f_{\mathcal{B}}^k(\bar{x} + d) + \tfrac{1}{2t} || d ||_2^2 \right\} \tag{28}$$

in our dual setting (3) is easily seen to have as dual

$$\sup \left\{ \sum_{k \in \mathcal{K}} (c^k - \bar{x} A^k) u^k - \tfrac{t}{2} \Big|\Big| \sum_{k \in \mathcal{K}} A^k u^k - b \Big|\Big|_2^2 \ : \ u^k \in U_{\mathcal{B}}^k \quad k \in \mathcal{K} \right\}$$

where $U^k_{\mathcal{B}} = co\, \mathcal{B}^k$; that is, to each component U^k of the feasible region (whose cartesian product makes U) is separately inner-approximated, as in (13)–(17). As we have seen in (18)–(21), however, for some type of sets U^k inner-approximation is *not* an efficient representation. We will in fact concentrate on sets where there is *no need of approximating anything*, because a "compact" representation *exists and is known* (cf. Y in (21)) that can be directly added to the master problem.

3.2 Exact models of "easy" components: the basic case

We now consider the special case of (2) with $\mathcal{K} = \{1, 2\}$

$$\sup \left\{ \; c^1 u^1 + c^2 u^2 \;:\; u^1 \in U^1 \;,\; u^2 \in U^2 \;,\; A^1 u^1 + A^2 u^2 = b \; \right\}$$

where the solution of the first Lagrangian subproblem

$$f^1(x) = \sup \left\{ \; (c^1 - xA^1) u^1 \;:\; u^1 \in U^1 \; \right\}$$

is only available through a standard "very opaque black box", however exotic it may be (e.g. [25]), whereas in the second Lagrangian subproblem

$$f^2(x) = \sup \left\{ \; (c^2 - xA^2) u^2 \;:\; u^2 \in U^2 \; \right\} \tag{29}$$

$U^2 = \{ \; u^2 \;:\; Gu^2 \leq g \; \}$ is an "easy" polyhedron; say, G is explicitly known and has "not too many" rows (think Y in §2.1). As already remarked, linearity is not necessary, and it is kept only for simplifying the results; also, the proposed method immediately extends to more than two components, as well as to other situations that will be explicitly discussed in the following.

The standard approach up to now has been to treat both components in exactly the same way: develop two (different) "black box" solvers, one for each component, compute a pair (u^1_+, u^2_+) at each iteration by invoking both on x_+, and add each to the respective bundle $\mathcal{B}^1$ and $\mathcal{B}^2$, thereby inner-approximating both U^1 (actually, its convex hull) and U^2. As we have seen in §2, however, this is something we do *not* want to do for U^2, since its "compact" representation via G and g is preferable. The solution is, on the outset, simple: form (28) where only f^1 is approximated by the cutting plane model $f^1_{\mathcal{B}}$, while f^2 is just kept as it is (since only one of the two functions is approximated there is no $\mathcal{B}^2$ and therefore we simply write $\mathcal{B}$ for $\mathcal{B}^1$). This, as anticipated in §2, is actually easy to do from the primal viewpoint: just carry forward the original U^2 constraints in the master problem, i.e., solve

$$\begin{array}{ll} \sup & \sum_{i \in \mathcal{B}} c^1_i \theta_i + c^2 u^2 + \bar{x} z - \frac{t}{2} || \, z \, ||_2^2 \\ & z = b - A^1 \left(\sum_{i \in \mathcal{B}} u^1_i \theta_i \right) - A^2 u^2 \quad , \quad G u^2 \leq g \quad , \quad \theta \in \Theta \end{array} \tag{30}$$

with the corresponding "abstract form" (for $U^1_{\mathcal{B}} = co\, \mathcal{B}$)

$$\sup \left\{ \; c^1 u^1 + c^2 u^2 + \bar{x} z - \tfrac{t}{2} || \, z \, ||_2^2 \;:\; z = b - A^1 u^1 - A^2 u^2 \;,\; u^1 \in U^1_{\mathcal{B}} \;,\; u^2 \in U^2 \; \right\}$$

(cf. again (18)–(21)), while keeping all the rest of the augmented Lagrangian approach unchanged. The dual of (30) can be easily verified to be

$$\begin{array}{ll} \inf & (\bar{x} + d) b + v^1 + g\omega + \frac{1}{2t} || \, d \, ||_2^2 \\ & v^1 \geq \left(c^1 - (\bar{x} + d) A^1 \right) u^1_i \qquad i \in \mathcal{B} \\ & \omega G = c^2 - (\bar{x} + d) A^2 \quad , \quad \omega \geq 0 \end{array} \tag{31}$$

where ω are the dual variables of the $Gu^2 \leq g$ constraints in (30). It is also easy to extract all useful information from this pair of dual quadratic problems: for instance, $\tilde{d}$ and $\tilde{v}^1 = f^1_{\mathcal{B}}(\bar{x} + \tilde{d})$ are the dual optimal multipliers of constraints $z = b - A^1 \left(\sum_{i \in \mathcal{B}} u^1_i \theta_i \right) - A^2 u^2$ and $\sum_{i \in \mathcal{B}} \theta_i = 1$, respectively, and so on. A small but useful remark is that in the above derivation the Lagrangian function actually has *three* terms: f^1, f^2, and $f^0(x) = xb$; indeed, (31) has the three corresponding terms v^1, $g\omega$, and $(\bar{x} + d)b$, respectively, in the objective function (apart from the stabilizing term). This fact, that we have purposely not discussed in the general derivation of §3.1, is interesting here because it shows that also in the normal development (and even with a non-sum f) one typically has at least one "very easy" component that is *not* approximated in the master problem. Similar results hold if the objective function $c^2(\cdot)$ and the constraint function $G(\cdot)$ are nonlinear but allow for closed-form duals, such as those expressible as SOCPs or SDPs [7], as well as under more general settings [27].

Clearly, little changes in the standard convergence theory of bundle methods. Only a few simple properties are required to models $f^k_{\mathcal{B}}$—basically not to overestimate f^k and to be "tight enough" at points where the function is computed [21, §5]—and the *real* function f^k cannot but satisfy those requirements. Usually, the crucial difference between f^k and $f^k_{\mathcal{B}}$ is that the former is either unknown or too complex to use, while the latter is efficiently implementable: as this is no longer true for "easy" components, it makes sense to use the original f^k as a model. Because the objective $f = f^0 + f^1 + f^2$ is overall a "difficult" function that needs to be approximated by $f_{\mathcal{B}} = f^0 + f^1_{\mathcal{B}} + f^2$, all the machinery of the bundle approach is still required, and not much actually changes in the outlook of the algorithm. Only, because some of the components are exactly known, the algorithm is provided with better information on the function to be minimized, and one expects faster convergence in practice.

Note that, provided that the function value is *only* computed at x_+, one can entirely avoid the oracle for the easy components. Indeed, once $f^1(x_+)$ is known, the value of $f(x_+)$ can be computed since $f^2(x_+)$ is obtained "for free" as a by-product of the solution of (31). This requires doing away with line—or curved [57]—searches; yet this is what happens anyway in most practical implementations, and the convergence theory requires computing $f(x_+)$ at least "frequently enough" [21]. A minor issue with this approach is that at the very first iteration—when (31) has never been solved yet—the value of $f(\bar{x})$ is not known, and therefore the descent condition cannot be checked; however, there are several easy workarounds for this, as the convergence theory allows to skip the descent test entirely "from time to time" [21].

Another relevant feature of bundle methods, *aggregation*, is not impacted at all from the use of easy components. This is based on the fact that the optimal solution $\tilde{u}$ of (26) "has just the right form to be added to $\mathcal{B}$"; under appropriate conditions [21] (incidentally, satisfied by (24) but not by (27)), doing so allows to prune down the bundle to any fixed size—downto $\mathcal{B} = \{ \tilde{u} \}$—without impairing convergence. A milder way to attain the same result is to keep in $\mathcal{B}$ all the u_i such that $\tilde{\theta}_i > 0$. Both these techniques can be easily adapted here: simply, one only aggregates on the "difficult" components—e.g., $\tilde{u}^1 = \sum_{i \in \mathcal{B}} u^1_i \tilde{\theta}_i$ in (30)—and prunes their bundles after having inserted the aggregated solution—downto $\mathcal{B} = \{ \tilde{u}^1 \}$—in (30). Clearly, no aggregation is possible and needed for the "easy" components. Yet, this allows to keep the maximum size of the master problems bounded, as the "easy" components result in a constant number of variables and constraints.

This basic idea readily extends to all the typical variants and tricks of standard bundle methods, as discussed below.

3.3 The constrained case

Bundle methods easily cope with constraints, provided the set X is known and (closed) convex, by simply inserting full knowledge about X into (24), i.e., solving

$$\inf \left\{ \; f_{\mathcal{B}}(\bar{x}+d) + \tfrac{1}{2t}||\, d\,||_2^2 \; : \; \bar{x}+d \in X \; \right\} \tag{32}$$

which is nothing but (24) using the model $f_{X,\mathcal{B}} = f_{\mathcal{B}} + I_X$ of the *essential objective* $f_X = f + I_X$. For polyhedral $X = \{\, x \in \mathbb{R}^n \; : \; Hx \leq h \,\}$ this is easy to incorporate in (31), yielding

$$\begin{array}{ll} \sup & \sum_{i\in\mathcal{B}} c_i^1\theta_i + c^2u^2 + \omega h + \bar{x}z - \frac{t}{2}||\, z\,||_2^2 \\ & z = b + \omega H - A^1\left(\sum_{i\in\mathcal{B}} u_i^1\theta_i\right) - A^2u^2 \\ & Gu^2 \leq g \quad , \quad \theta \in \Theta \quad , \quad \omega \geq 0 \end{array} \tag{33}$$

as it is easy to verify: the dual of (33) is simply (31) with the added constraint $Hd \leq h - H\bar{x}$. For instance, sign constraints $x \geq 0$, associated to inequality constraints in (2), simply give rise to *slack variables* ω in the constraint defining z. Clearly, handling constraints in this way is nothing but another case of "easy" component, since one is simply using a suitable "exact" model of I_X in the stabilized primal master problem. This requires a specific (minor) update of the convergence theory only because I_X is *not* finite-valued, as opposed to the f^ks [21, §9.1]. Actually, whenever the set of defining inequalities of X is finite, it is not even required that all of them be inserted in the master problem in advance: when an unfeasible x is probed, the black box should just return $f(x) = +\infty$ and some of the violated inequalities. This is useful e.g. in applications to combinatorial optimization if some of the sets U^k are *not* compact, and therefore the Lagrangian functions can indeed evaluate to $+\infty$. However, with $X^k = \mathit{dom}\; f^k$ for $k \in \mathcal{K}$, $f(x) = +\infty$ at any point x outside $X = \cap_{k\in\mathcal{K}}\; X^k$, no matter which and how many of the X^k it fails to belong to. Thus, there is no reason to "disaggregate X", too: constraints are "global" in nature, although in practice one may want to keep track of which component has generated a specific constraint, as the associated multipliers are those of *rays* of some U^k that may be needed to reconstruct the solution in the u-space. The inherent finiteness of the underlying representation makes it easy to ensure that the algorithm remains convergent, provided that minimal care is exercised in not removing information "too quickly" (the extreme, obvious case being never removing anything); see e.g. [24] for a similar situation.

Interestingly, for "simple" sets X an alternative approach has been proposed to lessen the cost of the master problem. The *Proximal-Projection* idea [40] is to exploit aggregation to replace the solution of the master problem with just one (or possibly more) step(s) of a block-descent approach. In a nutshell, the observation—in the notation of (33)—is that the optimal solution $(\tilde{\theta}, \tilde{\omega}, \tilde{u}^2)$ to (33) gives the optimal aggregated subgradient $\tilde{z} = \tilde{z}^0 + \tilde{z}^X + \tilde{z}^1 + \tilde{z}^2 = b + \omega^* H - A^1\tilde{u}^1 - A^2\tilde{u}^2$ (with $\tilde{u}^1$ obtained by $\tilde{\theta}$), each term of the sum being an (approximated) subgradient of the corresponding component f^0, I_X, f^1 and f^2, respectively. What one does is to alternatively replace some of the components with the linearizations provided by these (approximated) subgradients, just as aggregation does; except that aggregation is "permanent", while in this approach the original data is later restored. To mirror the approach of [40], one could first solve a master problem where the constraints $\bar{x}+d \in X$ in (32) are replaced

by a linearization using $\tilde{z}^X$ from the previous iteration. Then, a second master problem is solved where I_X is restored, but *all* the other components are replaced by their linearizations ($\tilde{z}^1$ and $\tilde{z}^2$) resulting from the first master problem (this does nothing to f^0, of course). This provides an approximated solution of the "true" master problem (32), which corresponds to one round of a block-Gauss-Seidel approach to (33) where first ω is kept fixed (to the optimal value $\tilde{\omega}$ of the previous iteration) and θ, u are allowed to vary, then θ, u are kept fixed and ω is allowed to vary. Since the results of [40] hold when f is a sum-function and the disaggregated model is used, they *a fortiori* hold if some components of f are "easy" and our approach is employed. This suggests that the Proximal-Projection approach could be extended to a specialized handling of the "easy" components in case they have an appropriate structure. In particular, assume that (33) with ω and θ fixed is easily solved with specialized approaches due to the structure of U^2 (which may be an "easy" set, think Y in §2.1). Then, one could envision a three-step block-Gauss-Seidel approach where at each step only one among θ, ω and u^2 is allowed to vary, the other two being fixed. Note that the same idea (called *Alternating Linearization*) has been applied already for yet another notion of "simple" component in [41], so convergence of such an approach could be relatively easy to obtain; this is however out of the scope of the present paper and it is left for future developments, especially since our computational results seem to present a strong case against approximation of the master problem (cf. §4.1).

3.4 Exploiting lower bounds

In practice, it may be the case that global lower bounds are known, either on some of the components ($-\infty < l^k \leq f^k$), or on the whole objective function ($-\infty < l \leq f$); it is then desirable to incorporate this valuable information in the master problems. For individual lower bounds, this is pretty straightforward. Of course, this is only relevant for the "difficult" f^1, since for the "easy" f^2 the bound is implicit anyway in the (already available) complete description of the function. Then, inserting l^1 in the model simply results in a new constraint $v \geq l^1$ in (31), and thus in a new variable ρ^1 in (30), with cost l^1, that does not appear in the definition of z but does in the constraint $\sum_{i\in\mathcal{B}} \theta_i + \rho^1 = 1$. An analogous trick works if *all* components of f are difficult: the constraint $v^1 + v^2 \geq l$ does the job, corresponding to a unique ρ variable, distinct from the "individual" ρ^1 and ρ^2, participating in both constraints $\sum_{i\in\mathcal{B}^k} \theta_i^k = 1$ for $k = 1, 2$. A similar idea also works when f^2 is "easy", as the master problem

$$\begin{array}{lll} (-l+) \ \inf & v + \frac{1}{2t}||\, d\,||_2^2 & \\ & v^1 \geq \big(c^1 - (\bar{x}+d)A^1\big)u_i^1 & i \in \mathcal{B} \\ & v \geq (\bar{x}+d)b + v^1 + gy & ,\ v \geq l \\ & \omega G = c^2 - (\bar{x}+d)A^2 & ,\ \omega \geq 0 \end{array} \tag{34}$$

clearly represents the model $\max\{\, f^0 + f^1 + f^2\,,\, l\,\}$. The primal side requires a little bit of interpretation: the dual of (34) is

$$\begin{array}{ll} \sup & \sum_{i\in\mathcal{B}} c_i^1\gamma_i + c^2\mu^2 - l(1-\rho) + \bar{x}z - \frac{t}{2}||\, z\,||_2^2 \\ & z = \rho b - A^1 \sum_{i\in\mathcal{B}} u_i^1\gamma_i - A^2\mu^2 \quad ,\quad \rho = \sum_{i\in\mathcal{B}} \gamma_i \\ & G\mu^2 \leq \rho g \qquad , \qquad \gamma \geq 0 \qquad , \qquad \rho \in [0,1] \end{array}$$

and it can be better understood by assuming that the global lower bound l of the Lagrangian function f is associated to one feasible solution $(\bar{u}^1, \bar{u}^2) \in U^1 \times U^2$ of (2),

i.e., such that $A^1\bar{u}^1 + A^2\bar{u}^2 = b$, of cost $c^1\bar{u}^1 + c^2\bar{u}^2 = l$. In the primal master problem, one is in fact dealing with the solution

$$u^1(\rho) = (1-\rho)\bar{u}^1 + \rho\mu^1 \qquad , \qquad u^2(\rho) = (1-\rho)\bar{u}^2 + \rho\mu^2$$

where

$$\mu^1 = \textstyle\sum_{i\in\mathcal{B}} u_i^1\gamma_i = \rho\sum_{i\in\mathcal{B}} u_i^1\theta_i = \rho u^1 \qquad (\gamma = \rho\theta) \qquad , \qquad \mu^2 = \rho u^2 \ ,$$

which satisfies the constraints $z = b - A^1u^1(\rho) + A^2u^2(\rho)$. Thus, if $\rho > 0$, one only needs to scale γ (μ^1) and μ^2 and by $1/\rho$ to retrieve the solution θ (u^1) and u^2 of (30); in fact, note that $\sum_{i\in\mathcal{B}}\theta_i = 1$ since $\sum_{i\in\mathcal{B}}\gamma_i = \rho$. We remark that this derivation is made particularly easy by the linearity assumptions in (29); the proof can indeed be extended to other classes of functions, such as all the SOCP- or SDP-representable ones (of which there are plenty of interesting applications in this setting, cf. e.g. [1] for the former or [34,52] for the latter) exploiting the fact that the *perspective* of a SOCP-representable function is SOCP-representable [7], but this requires further work as detailed in [27]. Incorporating individual bounds $f^1_{\mathcal{B}} \geq l^1$ in the above development is instead straightforward.

4 Computational results

We now apply the developed approach to FC-MMCF. Actually, we do this for *two* different variants of the formulation; other than the *weak formulation* (5)–(8) we also consider the *strong formulation* obtained by replacing (8) with

$$0 \leq w^k_{ij} \leq u^k_{ij}y_{ij} \qquad\qquad (i,j)\in A \ , \ k\in K \tag{35}$$

which is well-known to significantly improve the lower bound [13] at the cost of $|A|\cdot|K|$ extra constraints. In the Lagrangian approach the constraints (35) need also be relaxed with multipliers $\beta = [\ \beta^k_{ij}\ ]_{(i,j)\in A\ ,\ k\in K} \geq 0$, yielding

$$f(\alpha,\beta) = \left\{ \begin{array}{l} \min \displaystyle\sum_{(i,j)\in A}\Big[\ \sum_{k\in K}(c^k_{ij} + \alpha_{ij} + \beta^k_{ij})w^k_{ij} + \Big(\ f_{ij} - \alpha_{ij}u_{ij} - \sum_{k\in K}\beta^k_{ij}u^k_{ij}\ \Big)y_{ij}\ \Big] \\ \qquad\qquad (6)\ ,\ (8)\ ,\ (9) \end{array} \right.$$

(note that the constraints (8), redundant when (35) are present, are used to tighten up the relaxation). The corresponding (much) better lower bound is then paid with a (much) larger dual space which, as we will see, very significantly impacts on the computational cost of the problem.

For both functions, several possibilities exist when solving the corresponding maximization problem via a bundle method. These do not impact on how the function is computed, but rather on *how the information produced by the computation is used in the master problem*. In particular, one can have:

- a *fully aggregated* (FA) version a-la (10)–(12), where the sum-structure is ignored;
- a *partly disaggregated with easy y* (PDE) version, where f is the sum of *two* functions, one for W and the other for Y, the latter being treated as an "easy" one;
- a *disaggregated with difficult y* (DD) version a-la (13)–(17), with $|K|+1$ functions, all of them "difficult", one for each W^k and a single one for Y;
- a *disaggregated with easy y* (DE) version a-la (18)–(21), which is the same as the above but with the Y component treated as an "easy" one.

Note that a Fully Disaggregated version would exist that is analogous to DD except for the fact that Y is disaggregated as well; as discussed in §2.1 this is however basically the same as DE, and therefore it has not been tested.

4.1 Computational setting

We have implemented the proposed approach within a general-purpose `C++` bundle code developed by the first author and already used with success in several other applications [10,13,28]. The structure of the code allows to solve the Lagrangian dual with different approaches, such as Kelley's Cutting Plane method, several "quick and dirty" subgradient-like methods [4,15], and the bundle method. The latter in particular is "generalized" in the sense that the solution of the master problem is delegated to a separate software component under an abstract interface; this allows to test different solution algorithms, and different stabilizing terms, without affecting the main logic of the bundle approach. For our experiments we have used the specialized quadratic solver described in [20] for (24), which however only supports Fully Aggregated master problems. Therefore, all other options have been tested with the "boxstep" stabilization (27); all the LPs have been solved with `CPLEX 12.2`, while the Lagrangian relaxations have been solved with efficient Min-Cost Flow solvers from the `MCFClass` project (cf. e.g. [29]). All the algorithms have been coded in `C++`, compiled with GNU `g++ 4.4.5` (with `-O3` optimization option) and ran single-threaded on a server with multiple Opteron 6174 processors ("Magny-Cours", 12 cores, 2.2 GHz) each with with 32 GB of RAM, under a i686 GNU/Linux (Ubuntu 10.10 server). The computation of the Lagrangian function could have been easily parallelized [12], but, as the results will show, this would have hardly—if at all—improved the running times.

The experiments have been performed on 48 randomly generated problem instances. The random generator is the one already employed in several studies (e..g. [13]), but due to the remarkable effectiveness of the new approach we were able to tackle much larger instances than previously possible. In particular, we generated 12 groups of 4 instances each as follows. The number of nodes and arcs were chosen in the set $\{(20, 300), (30, 600), (50, 1200)\}$. For each of these, the number of commodities was chosen in the set $\{100, 200, 400, 800\}$. Then, each of the four instances with the same size differs for the parameters which control how "tight" the capacities are, and how "large" the fixed costs are. The characteristics of the 12 groups are summarized in Table 4.1; the instances can be freely downloaded from

`http://www.di.unipi.it/optimize/Data/MMCF.html#Canad` .

For the largest instances, the formulation has 960000 continuous variables and 1200 binary ones, 40000 equality constraints, and 962400 inequality constraints (not counting the sign restrictions). Of the latter, in the weak formulation 960000 are simple bound restrictions, whereas in the strong one they are the strong forcing constraints (35). While these are only slightly denser than bound restrictions, their impact on the bound computation time is quite dramatic, as the following results will show.

group	1	2	3	4	5	6	7	8	9	10	11	12
$\|N\|$	20	20	20	20	30	30	30	30	50	50	50	50
$\|A\|$	300	300	300	300	600	600	600	600	1200	1200	1200	1200
$\|K\|$	100	200	400	800	100	200	400	800	100	200	400	800

Table 1 Description of the instances

4.2 Results for the weak formulation

The results for the different approaches are presented in Table 2. In particular, the columns "FA-2" report results for the FA approach with (24), where the master problem is solved with the specialized quadratic solver of [20], while columns "FA-1" report results for the same approach with (27), where the master problem is solved with `Cplex`, as for all the other cases. For all the approaches, a maximum running time of 1000 seconds has been set, and the total running time (in seconds) is reported in column "time". The stopping criterion was set to a *relative* accuracy of `1e-6`, a rather high call for this kind of algorithms; this is implemented by asking that

$$c\tilde{u} + \bar{x}(\, b - A\tilde{u}\,) - \tfrac{t}{2}\big|\big|\, A\tilde{u} - b\,\big|\big|_2^2 \leq \varepsilon f(\bar{x}) \tag{36}$$

where $\varepsilon =$ `1e-6` and $\bar{t}$ is chosen large enough to ensure that $f(\bar{x}) - f^* \leq \varepsilon f(\bar{x})$ (where f^* is the optimal value of (1)) whenever (36) holds, which has been verified "ex post" in our experiments. To account for the case where such an accuracy is not reached within the allotted time, the final relative gap w.r.t. the "exact" lower bound computed with `Cplex` is reported in column "gap". The value is not reported if it is lower than `1e-12`, the default accuracy for the simplex-type algorithms in `Cplex`; if this always happens (as is the case with DE), the whole column is avoided. Column "iter" reports the number of iterations (computations of the Lagrangian function and master problem solutions). Finally, column "f" reports the total running time spent in the computation of the Lagrangian function; almost all the remaining time, which usually amounts to the vast majority, is spent in the master problem solution. The order of the rows in Table 2 is the same as that of the columns in Table 1, thus we avoid to report again the size of the instances in order to save on space; the results in each row are averaged among the four instances of the same group. The algorithmic parameters were tuned for all individual approaches, but uniformly on all instances; furthermore, the chosen sets of algorithmic parameters were, quite naturally, very similar to each other.

DE			PDE				DD				FA-1				FA-2			
time	f	iter	time	f	iter	gap	time	f	iter	gap	time	f	iter	gap	time	f	iter	gap
0.04	0.00	5	0.03	0.01	6		557	2.54	6200	1e-7	979	3.97	9105	1e-3	7.64	0.75	2383	1e-7
0.08	0.01	6	0.08	0.01	12		772	2.94	3153	6e-3	1000	4.43	4772	3e-2	14.24	1.37	1931	6e-9
0.25	0.01	7	0.57	0.12	52	1e-7	739	2.79	1365	2e-7	862	10.57	5579	3e-3	12.66	1.99	1117	5e-7
0.64	0.03	7	1.06	0.23	50	3e-7	1000	2.27	482	9e-3	1000	14.49	3201	8e-3	42.38	7.74	1714	7e-7
0.10	0.01	7	0.30	0.03	39		665	4.92	5799	4e-3	945	6.15	7538	8e-3	4.12	0.50	834	3e-7
0.25	0.02	10	1.81	0.21	122		498	3.37	1899	7e-8	808	9.76	5599	3e-3	6.36	1.06	664	1e-6
0.45	0.04	8	20.56	1.93	483	2e-7	1000	1.81	415	2e-2	1000	2.58	638	5e-2	134.49	15.00	3795	6e-7
1.10	0.08	9	5.17	1.09	120	1e-7	1000	3.48	378	2e-2	1000	10.08	1134	4e-2	126.29	26.19	2905	8e-7
0.34	0.02	11	34.80	0.78	449	5e-9	1000	1.39	746	5e-3	1000	2.23	1205	4e-2	28.92	2.77	1630	1e-6
0.42	0.05	9	2.39	0.26	89		1000	6.23	1647	3e-2	1000	8.51	2343	5e-2	32.77	5.26	1414	8e-7
0.99	0.10	11	16.03	2.34	271	1e-7	1000	6.18	717	2e-2	1000	11.31	1321	4e-2	80.05	16.48	1848	8e-7
2.19	0.18	10	124.38	13.95	811	6e-7	1000	5.05	278	2e-2	1000	14.63	838	6e-2	233.40	50.47	2851	8e-7

Table 2 Results for the weak formulation

Comparing FA-2 with FA-1, it is clear that the proximal stabilization (24) is in general largely preferable to the trust-region one (27). This has been reported several times over in different applications [6,11,24], and it just proves true once again here. Part of the result is due to the (still) more effective quadratic Master Problem solver of [20] w.r.t. general-purpose LP technology, as testified by the lower cost per iteration; however, the largest part of the improvement comes from faster convergence. Disaggregating W (DD) has surprisingly little effect; this is likely due to the fact that the master

problem cost is way higher (as testified by the yet higher iteration cost), without a sufficient effect on convergence speed to counterbalance it. Remarkably, "disaggregating Y alone" (PDE) has a far larger impact, being most often better than FA-2 despite the disadvantage of the trust-region stabilization. Yet, the decomposition method really shines when both W is disaggregated and Y is treated as an easy component (DE); this ends up being about two orders of magnitude faster than the best of the other approaches for the largest instances.

To put these results in the context of alternative available solution methods, in Table 3 we compare the running times of DE and FA-2 with these obtained by the several applicable LP algorithms in `Cplex`. These usually attain the much higher accuracy of `1e-12`, except barrier which usually falls more towards `1e-10`, and while DE (but not FA-2) actually obtains solution of comparable quality even when run with an accuracy of `1e-6`, there is a difference between reaching a solution with a given tolerance and being able to *certify* it. The latter typically involves producing a high-quality feasible primal solution (cf. (36)), which is therefore relevant to applications. Thus, for both Lagrangian approaches we report the running time required to stop when the accuracy is set to both `1e-6` and `1e-12`. The rows of the Table are arranged as those in Table 2.

`Cplex`						DE		FA-2	
primal	dual	barrier	p.net.	d.net.	auto	`1e-6`	`1e-12`	`1e-6`	`1e-12`
0.30	0.13	8.73	0.18	0.23	0.36	0.04	0.04	7.64	7.74
0.89	0.90	21.25	0.58	1.95	2.40	0.08	0.08	14.24	14.37
3.04	10.22	76.24	2.24	16.32	25.44	0.25	0.26	12.66	13.13
8.21	16.56	151.14	4.62	27.58	44.79	0.64	0.64	42.38	49.18
1.09	4.98	42.57	0.74	6.88	10.62	0.10	0.10	4.12	4.19
3.28	24.68	135.57	2.77	29.46	69.86	0.25	0.26	6.36	7.94
53.25	22.58	417.10	8.96	51.45	55.86	0.45	0.45	134.49	137.41
18.74	67.24	1115.22	10.56	99.96	177.40	1.10	1.10	126.29	163.88
19.98	84.33	303.29	3.92	112.71	187.37	0.34	0.35	28.92	42.71
7.89	82.64	583.52	18.60	259.65	309.74	0.42	0.42	32.77	40.60
38.09	230.79	1952.75	15.85	325.33	690.30	0.99	0.99	80.05	108.94
586.07	459.49	3586.63	51.71	738.23	1266.87	2.19	2.19	233.40	1789.08

Table 3 Comparison with `Cplex` for the weak formulation

The Table shows that, unlike what reported in [13], even at the lower accuracy setting FA-2 is not competitive with the best that `Cplex` offers, which is the primal network algorithm (somewhat surprisingly this is *not* automatically chosen, cf. the column "auto"); note that this refers to the fact that `Cplex` detects the partial network structure and solves a restricted network problem to provide a warm-start for the overall solution process, a strategy that can be extremely effective [23]. Furthermore, although the time required to obtain a very-high-accuracy solution of `1e-12` is usually not much different from that required to reach `1e-6`, there are cases where the difference is significant. It is not much surprising that the results of [13] are outdated now due to the giant strides in general-purpose LP technology since then; yet, said giant strides can be exploited with DE, whose master problem—which is where the vast majority of the time is spent—is solved with a general-purpose LP solver. Unlike the original formulation (5)–(9), the master problem is rather "unstructured", so that the specialized network-exploiting algorithms that make such a difference for `Cplex` are not poised to make any substantial contribution; indeed, `Cplex` algorithms perform much more similarly on the master problem than on the original formulation, with the "auto" setting now being perfectly appropriate. Yet, requiring the very-high accuracy of `1e-12` to DE hardly changes the required running time; thus, not only a high-quality solution is obtained

efficiently, but also its quality can be certified in basically the same time. Such high-quality solutions are obtained *at least* one order of magnitude faster than the best `Cplex` option, and several orders of magnitude faster than the others, on all but the smallest instances.

4.3 Results for the strong formulation

Due to the huge number of constraints (35), tackling the problem directly, either with a decomposition approach or with a LP solver, is not advisable. Rather, since one can expect that only a relatively small fraction of these constraints be actually active at optimality, one should resort to *dynamic generation*, whereby the constraints are only inserted in the formulation when they are found to be necessary. For `Cplex`, this is actually very simple: one only has to declare these as *lazy constraints*, and the solver then takes the entire burden of deciding how and when checking for their violation. One minor technical consequence is that the problem has to be declared as a Mixed-Integer Linear Program even if one only wants to solve the continuous relaxation, i.e., one does not declare the y variables to be integer. In fact, `Cplex` checks violation of lazy constraints only when an "integral" solution is generated, that is, a solution where all variables declared as integer actually have integer values; in our case, this means just any feasible continuous solution. A similar approach can be used for the decomposition approach, and it has already shown to be very effective [23, 28] for very-large-scale Lagrangian optimization. This is true here as well, as demonstrated by Table 4 which compares the running time of the static version with that of the dynamic version for both `Cplex` and DE (similar results hold for all the other ones). The results are only limited to the first four, smaller groups of instances (the first four columns in Table 1), and of course compare between identical settings for both approaches (the "auto" setting for `Cplex`), save for static vs. dynamic generation of the constraints.

`Cplex`		DE	
static	dynamic	static	dynamic
53.68	9.94	44.23	31.69
315.26	53.63	232.56	47.53
1539.23	113.91	1234.08	28.98
2788.91	458.01	2227.04	65.31

Table 4 Comparison of static and dynamic constraint handling

As the Table shows, the impact of dynamic generation is already very large for `Cplex`, reaching one order of magnitude; for the decomposition approach it is even more humongous, being close to two orders of magnitude (and actually far surpassing it for larger instances not shown here). Thus, in the following we will always use dynamic generation.

Due to the previous results, we should expect that not all the decomposition approaches be capable of solving the strong formulation efficiently enough. This is indeed the case, as shown in Table 5 again only for the four groups of smaller instances (things only get worse as size increases). The meaning of the rows and columns (where applicable) is the same as in Table 2. In order to try to compensate for the increase in size of the problem as the number of commodities grows—which now heavily impacts the number of Lagrangian multipliers, instead as "only" the cost of computing the Lagrangian function and possibly the number of columns in the master problem—the maximum running time is now dependent on $|K|$; in particular, it is fixed to 1000, 3000, 9000 and 27000 seconds when $|K|$ is 100, 200, 400 and 800, respectively.

DE			PDE			DD			FA-1			FA-2		
time	iter	gap	time	iter	gap	time	iter	gap	time	iter	gap	time	iter	gap
31.69	77	1e-7	1000	2980	2e-2	1000	2714	2e-1	1000	1990	2e-1	410.30	14880	9e-7
47.53	30	3e-7	3000	2896	6e-2	3000	3720	7e-2	3000	7351	2e-1	1854.97	11141	3e-6
28.98	24	2e-7	9000	8370	2e-2	9000	5061	5e-2	9000	10918	1e-1	1254.21	9035	2e-6
65.31	20	3e-8	27000	5618	3e-2	27000	2148	4e-2	27000	5293	8e-2	1732.17	12940	1e-6

Table 5 (Partial) results for the strong formulation

As the Table shows, even allowing for such long computing times most decomposition approaches do not even come close to the required `1e-6` precision. FA-2 actually succeeds almost always, failing to reach the required precision (and not by much) in only one of the 16 instances. This is due to two advantages: the quadratic stabilizing term which provides much better convergence, and the specialized master problem solver which allows it to perform many more iterations in the same time, thus giving it far better chances to get near to a good dual solution. Yet, DE is again by far the best approach, delivering solutions with the prescribed accuracy in a tiny fraction of the allotted time. Furthermore, DE can efficiently solve all the instances to much higher precision. This is shown in Table 6; since (as it could be expected by Table 5) getting to `1e-12` is more difficult than in the weak formulation case, we report the behavior of the algorithm for the four settings `1e-6`, `1e-8`, `1e-10` and `1e-12`. The meaning of the rows and columns is the same as in Table 2, except for the extra columns "add" which report the time required to dynamically check the violation of constraints (35). As in Table 2, "ex-post" gaps smaller than `1e-12` are not reported; since this always happens for `1e-12`, the corresponding column is avoided entirely. Also, columns "f" and "add" are avoided for the intermediate precisions to improve the readability of the Table; the trends closely follows these for the two extreme precisions.

`1e-6`					`1e-8`			`1e-10`			`1e-12`			
time	f	add	iter	gap	time	iter	gap	time	iter	gap	time	f	add	iter
31.69	0.05	0.96	77	1e-7	57.73	143	4e-9	62.07	170	3e-11	63.78	0.11	1.10	181
47.53	0.04	2.04	30	3e-7	51.22	33	2e-9	51.37	33		51.38	0.05	2.06	33
28.98	0.07	2.70	24	2e-7	29.15	25		29.15	25		29.16	0.07	2.74	25
65.31	0.14	6.58	20	3e-8	65.67	21		65.68	21		65.69	0.15	6.61	21
25.93	0.04	0.89	47	8e-8	28.28	51	3e-9	32.00	57		32.00	0.06	0.93	57
27.97	0.09	1.48	36	4e-7	55.43	51	4e-10	56.01	52	1e-11	56.28	0.12	1.60	52
20.80	0.09	1.80	21	2e-8	20.84	21	2e-9	25.69	24		25.69	0.11	1.84	24
132.60	0.24	10.03	23	8e-8	132.74	23		132.76	23		132.78	0.24	10.09	23
2.47	0.06	0.48	26	2e-10	2.47	26	2e-10	2.57	27	3e-12	2.66	0.06	0.49	27
245.91	0.26	4.18	59	1e-7	295.56	72	4e-9	333.22	84	2e-11	337.38	0.39	4.54	86
283.71	0.43	7.24	39	7e-8	442.56	55	2e-9	506.83	63	5e-12	507.52	0.71	7.78	63
241.84	0.52	11.85	24	2e-11	241.88	24	2e-11	241.92	24	2e-11	253.59	0.55	11.98	25

Table 6 Results for DE with varying precision

The picture painted by Table 6, albeit not as rosy as that of the corresponding Table 3 for the weak formulation, is still quite good: doubling the *certified* precision from `1e-6` to `1e-12` requires no more than doubling the running time, and much often far less. Note that e.g. for bounding purposes within an enumerative algorithm, `1e-6` is typically more than enough already.

Remarkably, in order to obtain these results it is instrumental to "let information accumulate". In particular, the best algorithmic settings for DE are:

- the maximum size of the bundle is set to $50 \cdot |K|$, and subgradients are only removed if their multiplier θ_i is zero for 40 consecutive iterations;
- constraint violation is checked at *every* iteration of the bundle algorithm, and constraints whose Lagrangian multiplier is zero are *never* removed.

These are pretty "extreme" settings which surprised us. The bundle size of 40000 in the largest instances would be absolutely unmanageable for the standard quadratic programming solvers [20] often used (not without a certain success, cf. FA-1 vs. FA-2 in Table 2 and 5) in bundle methods, and clearly requires access to state-of-the-art LP technology to be viable. Checking violation at every iteration is also uncommon; as a result, the time required for performing this function ("add" in Table 6) is rather large, actually much larger than that required to compute the Lagrangian function itself and as much as 10% of the total running time in some cases. However, the time for computing the Lagrangian function is a very small fraction of the total time, that is largely dominated by the master problem time, even more so than in the weak case; thus, the overall impact on performances is by far compensated by the very consistent improvement in the convergence speed. This is confirmed by Table 7, where we show the effect of even "slight" changes in these parameters on the performances of DE for the first four groups of instances (as in Table 5, and with the same maximum time). In particular, columns "opt" are the best settings used for the results in Table 5 and 6, columns "$20 \cdot |K|$" are relative to setting the maximum size of the bundle to $20 \cdot |K|$, columns "Rmv = 20" are relative to removing subgradients if their multiplier θ_i is zero for 20 (as opposed to 40) consecutive iterations, and columns "Sep = 10" are relative to performing separation of constraints (35) every 10 iterations (as opposed to every iteration).

opt				$20 \cdot \|K\|$				Rmv = 20				Sep = 10			
time	add	iter	gap	time	add	iter	gap	time	add	iter	gap	time	add	iter	gap
31.69	0.96	77	1e-7	289.41	2.27	841	7e-7	104.60	1.20	218	2e-7	72.96	1.35	194	1e-6
47.53	2.04	30	3e-7	3000.76	7.67	1585	3e-4	1564.82	4.99	803	4e-5	363.67	4.12	159	3e-7
28.98	2.70	24	2e-7	1125.93	6.73	726	4e-7	2585.05	7.82	796	1e-6	141.61	5.51	65	1e-6
65.31	6.58	20	3e-8	81.33	6.68	20	3e-8	17415.68	28.00	2121	8e-5	669.34	18.82	78	5e-7

Table 7 Effect of different algorithmic parameters settings on DE

The Table shows that curtailing information accumulation has in general dire consequences, although the details differ for the different parameters. For instance, requiring $|\mathcal{B}| \leq 20 \cdot |K|$ has little effect when $|K| = 800$, indicating that the optimal bundle dimension is likely to be somewhat sublinear in $|K|$, although the size of the master problem is linear in $|K|$. Conversely, the effect of "early removal" of subgradients (Rem = 20) grows dramatically as $|K|$ increases. Of course, it is likely that a tipping point exist after which disaggregation and aggressive accumulation of information becomes counterproductive, as the cost of the master problem increases faster than the improvement in convergence speed can compensate. This is well-known for instance in stochastic programming, even if the tipping point may be higher than what the folklore estimates [9] owing to improvement in LP technology. The same phenomenon is likely to occur for multicommodity problems, e.g. for the instances related to telecommunication problems where $|K| \in O(|N|^2)$. However, no sign of this even starting to happen has been detected in our test bed, even for $|K| = 800$.

To put again these performances in context, in Table 8 we compare DE (at the two accuracies `1e-6` and `1e-12`) with the various LP algorithms in `Cplex`, FA-2 and the FA model solved with the Volume algorithm (FA-V) [4]; subgradient-type approaches have often been found competitive with bundle ones for the approximate solution of difficult large-scale problems [13,28]. As a minor note, since the problem to be solved is now "formally" a MILP (to allow use of lazy constraints), there is no longer a way to specify primal or dual network simplex, but only a generic "network simplex". This does not look to be an issue, as in this case `Cplex` is much better at actually picking

the best solver: the automatic choice invariably reverts on the dual simplex (and thus need not be reported), which is indeed appropriate. Anyhow, the performances of the different approaches are much more similar than in the case of the weak formulation (a factor of two rather than more than an order of magnitude). We also mention that we experimented using the Volume algorithm to "warm-start" the bundle method, a trick that sometimes pays surprisingly good dividends [24]; unfortunately (but perhaps unsurprisingly), this was not one of these cases.

Cplex				DE		FA-2					FA-V				
primal	dual	net.	barr.	1e-6	1e-12	time	f	add	iter	gap	time	f	add	iter	gap
12	10	11	15	31.69	63.78	410	12	7	14880	9e-7	2.50	0.47	0.45	875	9e-3
64	53	61	71	47.53	51.38	1855	19	16	11141	3e-6	5.83	1.15	1.15	842	2e-2
139	114	132	157	28.98	29.16	1254	32	20	9035	1e-6	11.91	2.28	2.24	796	3e-2
559	456	531	587	65.31	65.69	1732	100	67	12940	1e-6	25.76	5.07	4.96	760	4e-2
46	39	43	60	25.93	32.00	322	12	10	10320	1e-6	5.53	0.88	1.13	871	8e-3
147	132	144	209	27.97	56.28	294	15	9	5300	1e-6	11.88	2.13	2.38	831	9e-3
509	301	478	648	20.80	25.69	5033	169	155	27231	1e-6	25.91	4.50	5.37	794	3e-3
2329	1930	2302	2590	132.60	132.78	3122	192	169	14547	1e-6	51.35	8.58	10.63	760	4e-2
196	131	156	304	2.47	2.66	344	20	12	7169	1e-6	11.61	1.99	2.30	827	3e-3
926	708	862	1174	245.91	337.38	2256	111	118	17034	2e-5	28.50	4.95	6.08	869	1e-2
2706	2167	2542	3272	283.71	507.52	5475	192	249	15061	3e-6	57.86	9.23	13.00	817	2e-2
11156	8908	11675	11683	241.84	253.59	11863	349	413	13953	1e-6	108.75	16.78	24.07	765	2e-2

Table 8 Comparison with `Cplex` and Volume for the strong formulation

The Table shows that FA-2 is capable of reaching a reasonably high accuracy of `1e-6` in almost all instances—but 5 out of 48, where anyway the ex-post accuracy is not too far from the desired target—within the allotted timeframe. This improves on [13], where bundle methods could only work with a very small maximum bundle size ($|\mathcal{B}| \approx 10$), and however could not produce very accurate solutions. The difference is to be mostly attributed to the dynamic generation of constraints, that was only implemented in a very primitive fashion in [13]. The Volume algorithm used here is remarkably more "robust" than the subgradient algorithms employed in [13]—bundle methods were found preferable mostly because much less dependent on fine-tuning of the algorithmic parameters—but, try as we might, we have never been able to have it attain more than `3e-3` precision. It is interesting to remark that for FA-2 separation is performed every 100 iterations. The Table clearly shows why this is necessary: even with this setting, the separation time ("add") is comparable to the function evaluation time ("f"). More frequent separation, say every 10 iterations, would render it far too costly. Remarkably, separation every 10 iteration is instead the best setting for the Volume algorithm, obtaining a reasonably low "add" time. This is due to the fact that the aggregated primal solution used to perform separation is obtained by the convex combination of only *two* solutions for FA-V, while many more solutions are used in FA-2; thus, computing the aggregated solution, rather than separation proper, is the costly operation. However, the FA methods are not competitive with `Cplex`: FA-V is faster than `Cplex` but obtains unacceptably coarse bounds, FA-2 is slower than `Cplex` while obtaining lower-quality bounds. Conversely, the DE bundle method obtains (extremely) accurate solutions much faster than `Cplex`, except for the smallest instances.

Qualitatively, the results can be described by saying that decomposition algorithms can work in "two different regimes". If enough information is collected and retained, they attain accurate optimal solutions in a few iterations, although with a high master problem cost (DE). If information is either aggregated (FA-2, FA-V) or withdrawn (Table 7), they tail-off rapidly, thus requiring many more iterations (and, usually, time) to reach the same precision. This shows that decomposition approaches highly benefit

from—indeed, require—very large master problems, which in turn call for appropriate solution technology. However, size is not the only factor at play here, as Table 5 clearly show, for otherwise DD should be roughly as successful as DE. The *structure* of the master problem, under the form of the appropriate compact representation (as opposed to the "entirely inappropriate" representation by its vertices) of the "easy" set Y, is at least as important. Thus, the "easy component" idea seems to be a key enabler, at least as far as the application considered in this paper goes.

5 Conclusions

Exploiting structure of the different components of the function to be minimized is crucial to construct efficient solution algorithms. This has repeatedly been shown in many contexts [48,49,50], and it appears to be a significant trend in the research about bundle-type methods for nondifferentiable optimization [2,24,34,40,41,45,56], delivering very promising results especially for Lagrangian relaxations of large-scale, highly structured problems. This work examines one form of structure which appears to be both quite general and widespread; hence, we expect several other (hopefully, successful) applications. Two possibilities come from two other staples of Lagrangian optimization: the hydro-thermal Unit Commitment problem in electrical power production and large-scale multistage stochastic programming problems. For the former, while thermal units require complex combinatorial oracles [25], hydro units—at least with some widely accepted simplifications—are just small-scale continuous flow problems, and therefore amenable to the "easy components" treatment to improve the performances of (already successful) Lagrangian approaches [3,10]. The latter also admits efficient Lagrangian approaches solved with subgradient [47], (inexact) bundle [18, 51], or augmented Lagrangian [55] methods: these could offer different targets to the "easy components" treatment, depending on which constraints are relaxed, such as the (usually) relatively few first-stage variables that play an analogous role to the design variables in FC-MMCF.

Our results also suggest several possible directions for future research. An interesting observation is that while these approaches can be very efficient, they require extremely large master problems which can be very costly to solve with general-purpose (even if state-of-the-art) LP technology. Besides, the cost for linear stabilization—at least with "simple" stabilizing functions, the story could be different with "complex" multi-pieces ones [6]—is a considerable decrease in the convergence speed w.r.t. quadratic stabilization. Thus, research should probably be resumed on specialized quadratic programming solvers capable of exploiting the structure of the master problems to substantially improve the performances; that the 15-years-old implementation of [20] is still effective nowadays is comforting in this respect, showing how powerful appropriate structure exploitation can be. Yet, for the approach proposed in this paper one would require a solver that on one hand is specialized enough to exploit the standard structure of the master problem as [20], but on the other hand is generic enough to be able to deal with as many structures of the "easy" components as possible, which seems a contradictory requirement. The Alternating Linearizaton approach (cf. §3.3) may be a direction to explore, but the task may call for the development of some entirely new solution methodologies for structured quadratic programms.

Acknowledgements

We are grateful to the anonymous Referees and to the Editor for their insightful comments. The first author gratefully acknowledges financial support of the Italian Min-

istry of Education, University and Research (MIUR) under grant PRIN 2009XN4ZRR. The second author thanks the financial support of European Commission through the European Social Fund and of the Regione Calabria under grant POR Calabria FSE 2007/2013.

References

1. A. Astorino, A. Frangioni, M. Gaudioso, and E. Gorgone. Piecewise Quadratic Approximations in Convex Numerical Optimization. *SIAM Journal on Optimization*, 21(4):1418–1438, 2011.
2. F. Babonneau and J.-P. Vial. ACCPM with a Nonlinear Constraint and an Active Set Strategy to Solve Nonlinear Multicommodity Flow Problems. *Mathematical Programming*, 120(1):179–210, 2009.
3. L. Bacaud, C. Lemaréchal, A. Renaud, and C. Sagastizábal. Bundle Methods in Stochastic Optimal Power Management: A Disaggregated Approach Using Preconditioners. *Computational Optimization and Applications*, 20:227–244, 2001.
4. L. Bahiense, N. Maculan, and C. Sagastizábal. The Volume Algorithm Revisited: Relation with Bundle Methods. *Mathematical Programming*, 94(1):41–70, 2002.
5. A. Beck and M. Teboulle. Mirror Descent and Nonlinear Projected Subgradient Methods for Convex Optimization. *Operations Research Letters*, 31:167–175, 2003.
6. H. Ben Amor, J. Desrosiers, and A. Frangioni. On the Choice of Explicit Stabilizing Terms in Column Generation. *Discrete Applied Mathematics*, 157(6):1167–1184, 2009.
7. A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, Engineering Applications*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2001.
8. D.P. Bertsekas and A. Nedić. Incremental Subgradient Methods for NonDifferentiable Optimization. *SIAM J. on Optimization*, 12(1):109–138, 2001.
9. J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, New York, 1997.
10. A. Borghetti, A. Frangioni, F. Lacalandra, and C.A. Nucci. Lagrangian Heuristics Based on Disaggregated Bundle Methods for Hydrothermal Unit Commitment. *IEEE Transactions on Power Systems*, 18(1):313–323, February 2003.
11. O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of Bundle and Classical Column Generation. *Mathematical Programming*, 113(2):299–344, 2008.
12. P. Cappanera and A. Frangioni. Symmetric and Asymmetric Parallelization of a Cost-Decomposition Algorithm for Multi-Commodity Flow Problems. *INFORMS Journal on Computing*, 15(4):369–384, 2003.
13. T.G. Crainic, A. Frangioni, and B. Gendron. Bundle-based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design Problems. *Discrete Applied Mathematics*, 112:73–99, 2001.
14. T.G. Crainic, B. Gendron, and G. Hernu. A Slope Scaling/Lagrangean Perturbation Heuristic with Long-Term Memory for Multicommodity Capacitated Fixed-Charge Network Design. *Journal of Heuristics*, 10(5):525–545, 2004.
15. G. d'Antonio and A. Frangioni. Convergence Analysis of Deflected Conditional Approximate Subgradient Methods. *SIAM Journal on Optimization*, 20(1):357–386, 2009.
16. O. du Merle, J.-L. Goffin, and J.-P. Vial. On Improvements to the Analytic Center Cutting Plane Method. *Computational Optimization and Applications*, 11(1):37–52, 1998.
17. G. Emiel and C. Sagastizábal. Incremental-like Bundle Methods with Application to Energy Planning. *Computational Optimization and Applications*, 46:305–332, 2010.
18. C.I. Fabian and Z. Szoke. Solving Two-Stage Stochastic Programming Problems with Level Decomposition. *Computational Management Science*, 4(4):313–353, 2007.
19. S. Feltenmark and K. Kiwiel. Dual Applications of Proximal Bundle Methods, including Lagrangian Relaxation of Nonconvex Problems. *SIAM Journal on Optimization*, 10(3):697–721, 2000.
20. A. Frangioni. Solving Semidefinite Quadratic Problems Within Nonsmooth Optimization Algorithms. *Computers & Operations Research*, 21:1099–1118, 1996.
21. A. Frangioni. Generalized Bundle Methods. *SIAM Journal on Optimization*, 13(1):117–156, 2002.

22. A. Frangioni. About Lagrangian Methods in Integer Optimization. *Annals of Operations Research*, 139:163–193, 2005.
23. A. Frangioni and G. Gallo. A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min Cost Flow Problems. *INFORMS Journal on Computing*, 11(4):370–393, 1999.
24. A. Frangioni and B. Gendron. A Stabilized Structured Dantzig-Wolfe Decomposition Method. *Mathematical Programming*, to appear, 2013.
25. A. Frangioni and C. Gentile. Solving Nonlinear Single-Unit Commitment Problems with Ramping Constraints. *Operations Research*, 54(4):767 – 775, 2006.
26. A. Frangioni and C. Gentile. Experiments with a Hybrid Interior Point/Combinatorial Approach for Network Flow Problems. *Optimization Methods and Software*, 22(4):573 – 585, 2007.
27. A. Frangioni and E. Gorgone. Generalized Bundle Methods for Sum-Functions with "Easy" Components: Applications to Multicommodity Network Design. Technical Report 12-12, Dipartimento di Informatica, Università di Pisa, 2012.
28. A. Frangioni, A. Lodi, and G. Rinaldi. New Approaches for Optimizing Over the Semimetric Polytope. *Mathematical Programming*, 104(2-3):375–388, 2005.
29. A. Frangioni and A. Manca. A Computational Study of Cost Reoptimization for Min Cost Flow Problems. *INFORMS Journal on Computing*, 18(1):61–70, 2006.
30. I. Ghamlouche, T.G. Crainic, and M. Gendreau. Path Relinking, Cycle-Based Neighborhoods and Capacitated Multicommodity Network Design. *Annals of Operations Research*, 131(1-4):109–133, 2004.
31. I. Ghamlouche, T.G. Crainic, and B. Gendron. Cycle-Based Neighborhoods for Fixed-Charge Capacitated Multicommodity Network Design. *Operations Research*, 51(4):655–667, 2003.
32. J.-L. Goffin and J.-P. Vial. Convex Nondifferentiable Optimization: a Survey Focussed on the Analytic Center Cutting Plane Method. *Optimization Methods and Software*, 17(5):805–867, 2002.
33. J. Gondzio, P. González-Brevis, and P. Munari. New Developments in the Primal-dual Column Generation Technique. Technical Report ERGO-11-001, School of Mathematics, The University of Edinburgh, 2011.
34. C. Helmberg and F. Rendl. A Spectral Bundle Method for Semidefinite Programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.
35. J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms* II—*Advanced Theory and Bundle Methods*, volume 306 of *Grundlehren Math. Wiss.* Springer-Verlag, New York, 1993.
36. K. Holmberg and D. Yuan. A Lagrangian Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem. *Operations Research*, 48(3):461–481, 2000.
37. K.L. Jones, I.J. Lustig, J.M. Farwolden, and W.B. Powell. Multicommodity Network Flows: The Impact of Formulation on Decomposition. *Mathematical Programming*, 62:95–117, 1993.
38. K. Kiwiel and C. Lemaréchal. An Inexact Bundle Variant Suited to Column Generation. *Mathematical Programming*, 118:177–206, 2009.
39. K.C. Kiwiel. A Bundle Bregman Proximal Method for Convex Nondifferentiable Optimization. *Mathematical Programming*, 85:241–258, 1999.
40. K.C. Kiwiel. A Proximal-Projection Bundle Method for Lagrangian Relaxation, Including Semidefinite Programming. *SIAM Journal on Optimization*, 17(4):1015–1034, 2006.
41. K.C. Kiwiel. An Alternating Linearization Bundle Method for Convex Optimization and Nonlinear Multicommodity Flow Problems. *Mathematical Programming*, 130(1):59–84, 2011.
42. G. Kliewer and L. Timajev. Relax-and-Cut for Capacitated Network Design. In *Algorithms – ESA 2005*, volume 3669/2005 of *Lecture Notes in Computer Science*, pages 47–58. Springer, Berlin / Heidelberg, 2005.
43. C. Lemaréchal. Lagrangian Relaxation. In M. Jünger and D. Naddef, editors, *Computational Combinatorial Optimization*, pages 115–160. Springer-Verlag, Heidelberg, 2001.
44. C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69:111–147, 1995.
45. C. Lemaréchal, A. Ouorou, and G. Petrou. A Bundle-type Algorithm for Routing in Telecommunication Data Networks. *Computational Optimization and Applications*, 44(3):385–409, 2009.
46. C. Lemaréchal and A. Renaud. A Geometric Study of Duality Gaps, with Applications. *Mathematical Programming*, 90:399–427, 2001.

47. A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust Stochastic Approximation Approach to Stochastic Programming. *SIAM Journal on Optimization*, 19:1574–1609, 2009.
48. A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*, volume XV of *Wiley-Interscience Series in Discrete Mathematics*. John Wiley, 1983.
49. Y. Nesterov. Smooth Minimization of Non-smooth Functions. *Mathematical Programming*, 103(1):127–152, 2005.
50. Y. Nesterov. Barrier Subgradient Method. *Mathematical Programming*, to appear, 2013.
51. W. Oliveira, C. Sagastizábal, and S. Scheimberg. Inexact Bundle Methods for Two-Stage Stochastic Programming. *SIAM Journal on Optimization*, 21(2):517–544, 2011.
52. M.R. Oskoorouchi and J.-L. Goffin. The Analytic Center Cutting Plane Method with Semidefinite Cuts. *SIAM Journal on Optimization*, 13(4):1029–1053, 2003.
53. A. Ouorou. A Proximal Cutting Plane Method Using Chebychev Center for Nonsmooth Convex Optimization. *Mathematical Programming*, 119(2):239–271, 2009.
54. A. Ouorou. The Proximal Chebychev Center Cutting Plane Algorithm for Convex Additive Functions. *Mathematical Programming*, to appear, 2013.
55. A. Ruszczynski. *Decompostion methods*, volume 10 of *Handbooks in Operations Research and Management Science*, pages 141–211. Elsevier, 2003.
56. C. Sagastizábal. Composite Proximal Bundle Method. *Mathematical Programming*, to appear, 2013.
57. H. Schramm and J. Zowe. A version of the Bundle Idea for Minimizing a Nonsmooth Function: Conceptual Idea, Convergence Analysis, Numerical Results. *SIAM Journal on Optimization*, 2(1):121–152, 1992.