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

Antonio Frangioni
Dipartimento di Informatica, Università di Pisa
Polo Universitario della Spezia
Via dei Colli 90, 19121 La Spezia (SP) – Italy
frangio@di.unipi.it

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

Abstract

We propose a modification to the (generalized) bundle scheme for minimization of a convex nondifferentiable sum-function in the case where some of the components are "easy", that is, they are Lagrangian functions of explicitly known convex programs with "few" variables and constraints. This happens in many practical cases, particularly within applications to combinatorial optimization. In this case one can insert in the master problem a suitably modified representation of the original convex subproblem, as an alternative to iteratively inner-approximating it by means of the produced extreme points, thus providing the algorithm with exact information about a part of the objective function, possibly leading to performance improvements. This is shown to happen 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, Partial Dantzig-Wolfe Decomposition, Multicommodity Network Design*

1 Introduction, motivation

We are concerned with the numerical solution of the problem

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

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. *Bundle methods* are known to be among the most efficient implementable algorithms for this class of problems. We will refer in particular to the *generalized* ones defined in [13], since that is perhaps the largest class of bundle methods with a unified convergence analysis. However, several other bundle algorithms have been proposed (see [24] for a through discussion and some more recent references like [27, 28, 29, 32, 34]), and the basic idea of the present paper appears to be easily applicable to most of them as well. A particularly relevant application of these approaches is the one where f is a *Lagrangian function* of some optimization problem; some *linking constraints* are relaxed so that the Lagrangian relaxation decomposes into $|\mathcal{K}|$ independent subproblems, and the minimization of f corresponds to the solution of the Lagrangian dual. In this case, it is known that bundle algorithms also solve a "convexified" version of the original problem [13, 11, 33], which is useful e.g. in combinatorial optimization [14].

It is standard in bundle methods to assume that f is only known through an oracle ("black box") that, given any $x \in X$, returns the value $f(x)$ and $z \in \partial f(x)$. To simplify the treatment, we will initially assume $X = \mathbb{R}^n$, with the extension to the constrained case to be separately discussed later on. Bundle methods generate a sequence of *tentative points* $\{x_i\}$ where the oracle is probed to gather the *bundle* $\mathcal{B} = \{ \ (\ f(x_i) \ , \ z_i \in \partial f(x_i) \ ) \ \}$; in our setting, the typical assumption is that a separate oracle is available for each component providing $f^k(x_i)$ and $z_i^k \in \partial f^k(x_i)$, so that $f(x_i) = \sum_{k \in \mathcal{K}} f^k(x_i)$ and $z_i = \sum_{k \in \mathcal{K}} z_i^k$. These are used to construct a *model* $f_{\mathcal{B}}$ of f; typically, the (aggregated) *cutting-plane model*

$$\hat{f}_{\mathcal{B}}(x) \ = \ \max \left\{ \ f(x_i) + z_i(x - x_i) \ : \ i \in \mathcal{B} \ \right\} \ \leq \ f \ . \tag{2}$$

This is fundamental to construct the next tentative point x_+; its simplest choice—that of Kelley's Cutting Plane method—is just a minimum of $\hat{f}_{\mathcal{B}}$, but some form of *stabilization* (discussed below) is usually necessary in order to improve the performances. Several different forms of stabilization have been analyzed in the literature, but the focus of the present paper is rather the choice of the model $f_{\mathcal{B}}$. It has been repeatedly shown that exploiting as much as possible the *structure* of the problem at hand is highly beneficial. For a sum-function, a relatively straightforward idea is to separately keep the $|\mathcal{K}|$ *disaggregated bundles* $\mathcal{B}^k = \{ \ (\ f^k(x_i) \ , \ z_i^k \ ) \ \}$. Doing so allows to construct $|\mathcal{K}|$ *independent cutting plane 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 bundle $\mathcal{B} = \cup_{k \in \mathcal{K}} \mathcal{B}^k$—of f than (2). This more accurate *disaggregated cutting plane model* improves the convergence speed so much as to amply compensate the increased cost due to the larger size of the corresponding master problems (cf. (13)–(14) below) [2, 5, 26].

All this has been known for a long time, but only recently it has been realized that in several applications not all the components of f are equivalent. Indeed, it is often the case that while some of the them do require a complex oracle, others have a "simple" form. Recent results—not by chance, motivated by a multicommodity flow structure similar to the one of interest here—have focussed on the case where one of the f^k is smooth [32, 29]. Furthermore, it can be argued (cf. §2.3) that even seemingly unrelated work on faster constrained bundle methods [28] basically hinges on the concept that the true objective function is the sum of two rather different components: a generic f, and the indicator function I_X of a "simple" set. In this paper, we will show a different approach along the same lines for the case where some of the components can be expressed by means of a "simple" convex program. This allows to use an *exact* model of these components by basically just copying the corresponding constraints in the formulation of the master problem, instead of approximating them piecemeal by inner linearization. This may enlarge the initial size of the master problem, but not necessarily the average size since the exact model does not grow in size with the iterations like classical ones do. Furthermore, it provides the algorithm with a better—indeed, "perfect"—knowledge of a part of the function to be minimized, thereby possibly improving the convergence speed. From the primal viewpoint, this amounts at performing a *Stabilized Partial Dantzig-Wolfe Decomposition* where not all the polyhedra whose Cartesian product forms the feasible region are reformulated be extreme points and iteratively inner approximated. The new approach leads to a very substantial reduction in the overall running time in at least one relevant application: the computation of tight lower bounds for Fixed-Charge Multicommodity Min-Cost Flow problems.

The structure of the paper is as follows. In Section 2 an overview of (generalized) bundle methods is provided, and we show how "exact" models of "easy" components can be inserted in the master problem. Then, in Section 3 the computational results obtained with Fixed-Charge Multicommodity Min-Cost Flow problems are presented and discussed, and, finally, in Section 4 conclusions are drawn.

Throughout the paper the following standard notation is used. Given a set X, $I_X(x) = 0$ if $x \in X$ (and $+\infty$ otherwise) is its *indicator function*, $\sigma_X(z) = \sup_x\{\ zx\ :\ x \in X\ \}$ is its *support function*, *cl* X is its *closure*, and *co* X is its *convex hull*. Given a convex function f, $\partial f(x)$ is its *subdifferential* at x, *epi* $f = \{\ (v,x)\ :\ v \geq f(x)\ \}$ is its *epigraph*, *dom* $f = \{\ x\ :\ f(x) < \infty\ \}$ is its *domain*, and $f^*(z) = \sup_x\{\ zx - f(x)\ \}$ is its *Fenchel's conjugate*. Given a problem (P) $\inf[\sup]_x\{\ f(x)\ :\ x \in X\ \}$, $v(P)$ denotes its optimal value; $X = \emptyset \Rightarrow v(P) = +[-]\infty$.

2 Bundle methods with "easy" components

We first provide an overview of (generalized) bundle methods in the "unstructured" case where f is any convex function provided by an oracle; then, we show what modifications are needed to the approach (mainly, to the master problem) in order to accommodate "exact" representation of the "easy" components of f.

2.1 A brief overview of (generalized) Bundle methods

Bundle methods are best described as implementable forms of (generalized) *Proximal Point* algorithms, i.e., as the combination of two different ideas:

Proximal Point In order to choose the next iterate f, one selects a *current point* $\bar{x}$ (usually the best iterate found so far) and a *stabilizing term* D_t—a closed convex function with suitable properties—and solves the *stabilized primal problem*

$$(\Pi_{\bar{x},t}) \qquad \phi_t(\bar{x}) = \inf_d \left\{\ f(\bar{x} + d) + D_t(d)\ \right\} \quad . \tag{3}$$

This amounts at computing the (generalized) *Moreau–Yosida regularization* ϕ_t of f in $\bar{x}$; with a proper D_t, ϕ_t has the same set of minima as f but enjoys additional properties, e.g., smoothness, making it easier to construct descent algorithms for its minimization. Under mild assumptions on D_t, the optimal solution d^* to (3) is 0 if $\bar{x}$ is optimal for (1), while d^* is a descent direction (that is, $f(\bar{x} + d^*) < f(\bar{x})$) otherwise.

Approximation Unfortunately, solving (3) with the sole help of the black box for f is, in principle, as difficult as solving (1). Therefore, bundle methods resort to a two-level approach, repeatedly solving the *stabilized primal master problem*

$$(\Pi_{\mathcal{B},\bar{x},t}) \qquad \phi_{\mathcal{B},t}(\bar{x}) = \inf_d \left\{ f_{\mathcal{B}}(\bar{x}+d) + D_t(d) \right\} \quad , \tag{4}$$

an approximation of (3) where $f_{\mathcal{B}}$ is a *model* of f constructed using the information available in $\mathcal{B}$. The optimal solution d^* of (4)—an approximation to the optimal solution of (3)—is used to compute the next iterate $x_+ = \bar{x} + d^*$. If the model is "accurate enough", d^* will indeed 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*) and convergence is attained as in the Proximal Point approach. Otherwise $\bar{x}$ is left unchanged (a *Null Step*) and some $z_+ \in \partial f(x_+)$ is added to $\mathcal{B}$ in order to improve $f_{\mathcal{B}}$. Sequences of consecutive Null Steps are to be seen as steps of an approach for the (approximated) solution of (3) based on iteratively refining the model $f_{\mathcal{B}}$ using information provided by the oracle. This approach may (at least asymptotically) provide the optimal solution of (3); however, for the sake of efficiency it is better to stop it as soon as the obtained d^* provides a large enough descent, which is the rationale for solving (3) in the first place.
In this process, D_t may be simply kept fixed (assuming that it has suitable properties, otherwise one has to ensure that eventually the stabilizing term vanishes [13, §8]), but on-line tuning of t 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. [31]). Anyway, general rules can be given [13] such that any t-strategy obeying them eventually constructs a minimizing sequence for (1).

A relevant aspect of bundle methods in the present context is their *dual* viewpoint. This hinges on the well-known concept of *Fenchel conjugate* of f, the closed convex function $f^*(z)$ which characterizes the set of all vectors z that are support hyperplanes to the epigraph of f. Among the several useful properties of f^* [13, 23], we just remind that from the very definition $f^*(0) = -v(\Pi)$; thus, the (apparently weird) dual problem

$$(\Delta) \qquad \inf_z \left\{ f^*(z) \ : \ z = 0 \right\} \tag{5}$$

is equivalent to (Π), i.e., $v(\Pi) = -v(\Delta)$. Likewise, one can thus construct the *Fenchel's dual* of any convex program by computing the conjugate of the objective function and evaluating it in 0; doing this for (4) reveals the *stabilized dual master problem*

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \inf_z \left\{ f^*_{\mathcal{B}}(z) - z\bar{x} + D^*_t(-z) \right\} \quad . \tag{6}$$

Clearly, (6) with $f_{\mathcal{B}} = f$ is the dual of (3), and can be seen as a (generalized) *augmented Lagrangian* of (5) where the constraints $z = 0$ are replaced with the linear term $-\bar{x}z$ (with Lagrangian multipliers $\bar{x}$) and the nonlinear term $D^*_t(-z)$ in the objective function. This becomes more revealing by realizing that the Lagrangian relaxation of (5) with respect to the constraints $z = 0$, using x as Lagrangian multipliers,

$$(\Delta_x) \qquad f(x) = -\inf_z \left\{ f^*(z) - zx \right\} \tag{7}$$

can be seen as the problem that the oracle has to solve for computing $f(x)$. Thus, bundle methods can also be seen as an approximated augmented Lagrangian approach to (5): (6) is solved and x is updated using the corresponding first-order information provided that $f^*_{\mathcal{B}}$

is an accurate enough model of f^*. All this becomes clearer when $f_{\mathcal{B}}$ is the cutting plane model (2), which in our notation is better rewritten

$$\hat{f}_{\mathcal{B}}(x) = \sup_{\bar{z}} \left\{ \, \bar{z}x - f^*(\bar{z}) \; : \; \bar{z} \in \mathcal{B} \, \right\} \;\; ,$$

(using $f(x_i) + f^*(z_i) = x_i z_i$ as $z_i \in \partial f(x_i)$). Thus

$$\hat{f}^*_{\mathcal{B}}(z) = \inf_{\theta} \left\{ \, \sum_{\bar{z} \in \mathcal{B}} f^*(\bar{z})\theta_{\bar{z}} \; : \; \sum_{\bar{z} \in \mathcal{B}} \bar{z}\theta_{\bar{z}} = z \;,\; \theta \in \Theta \, \right\} \quad ,$$

where $\Theta = \{ \, \sum_{\bar{z} \in \mathcal{B}} \theta_{\bar{z}} = 1 \;,\; \theta \geq 0 \, \}$ is the unitary simplex. In this case, the stabilized primal problem (3) is approximately solved by means of a simple cutting plane algorithm [24, Algorithm XII.4.2.1], where the unknown f is replaced with its known polyhedral outer approximation $\hat{f}_{\mathcal{B}}$. Correspondingly, the stabilized dual problem is approximately solved by means of an inner approximation approach, where the unknown f^* is replaced with its known polyhedral inner approximation $\hat{f}^*_{\mathcal{B}}$ (note that *dom* $\hat{f}^*_{\mathcal{B}} =$ *co* $\mathcal{B}$), the *dual pricing* problem (7) providing new pairs $(f^*(z_+), \; z_+)$ to improve the inner approximation.

Thus, every bundle algorithm can be read as an approach to solving (5), and one can show that, under fairly general assumptions, an optimizing sequence for the problem is constructed. This may appear highly uninteresting at first (after all, the only solution can be $z = 0$!) but this impression reveals itself wrong at least for the case where f is a Lagrangian function of a (convex) problem. That is, let

$$(\Omega) \qquad \sup_u \left\{ \, c(u) \; : \; Au = b \;,\; u \in U \, \right\} \quad , \tag{8}$$

and consider its Lagrangian function

$$(\Omega_x) \qquad f(x) = \sup_u \left\{ \, c(u) + x(b - Au) \; : \; u \in U \, \right\} \;\; . \tag{9}$$

It is well known that $f(x)$ is intimately tied with the (opposite of the) *value function* of (8)

$$\nu(z) = -\sup_u \left\{ \, c(u) \; : \; b - Au = z \;,\; u \in U \, \right\} \;\; ; \tag{10}$$

indeed,

$$\begin{aligned} \nu^*(x) &= \sup_z \left\{ \, zx + \sup_u \left\{ \, c(u) \; : \; b - Au = z \;,\;\; u \in U \, \right\} \right\} \\ &= \sup_u \left\{ \, c(u) + x(b - Au) \; : \; u \in U \, \right\} \;=\; f(x) \end{aligned}$$

for z can only take the value $b - Au$, making the sup irrelevant. Of course, this does not imply that $f^* = \nu$, as without assumptions on $c()$ and U we cannot even expect ν to be convex; rather, $f^* = \nu^{**}$, i.e., the closed convex envelope of ν. Indeed, it is well known that $v(\Pi) - v(\Omega) = \nu(0) - \nu^{**}(0)$, i.e., the duality gap between the original problem and its Lagrangian dual, if any, is entirely captured by the difference between ν and its convex envelope in 0 [31]. Thus, for $f^* = \nu$ to hold one has first to assume that (Ω) is convex in the first place—$c()$ is concave and U is convex—plus some technical assumptions that will be discussed later on. When this happens, plugging $f^* = \nu$ into the stabilized dual problem ((6) with $f_{\mathcal{B}} = f$) gives

$$(\Omega_{\bar{x},t}) \qquad \sup_u \left\{ \, c(u) + \bar{x}(b - Au) - D^*_t(Au - b) \; : \; u \in U \, \right\}$$

and optimal solutions of (5) naturally translate into very interesting objects. Indeed, if e.g. c is affine and the cutting plane model $\hat{f}_{\mathcal{B}}$ is used, (6) becomes

$$(\Omega_{\mathcal{B},\bar{x},t}) \quad \inf_{z,\theta} \left\{ \, -c\left(\sum_{\bar{u} \in \mathcal{B}} \bar{u}\theta_{\bar{u}}\right) - z\bar{x} + D^*_t(-z) \; : \; A\left(\sum_{\bar{u} \in \mathcal{B}} \bar{u}\theta_{\bar{u}}\right) - b = z \;,\; \theta \in \Theta \, \right\}$$

$$= \sup_u \{\ c(u) + \bar{x}(b - Au) - D_t^*(Au - b)\ :\ u \in co\,\mathcal{B} = U_{\mathcal{B}}\ \}\ , \tag{11}$$

where $\mathcal{B}$ is now seen as the set of optimal solutions $\bar{u} \in U$ of the dual pricing problem (9) such that $\bar{z} = b - A\bar{u}$. Thus, (generalized) bundle method combine an inner linearization approach (where U is substituted by its "simple" approximation $U_{\mathcal{B}}$) and a (generalized) augmented Lagrangian scheme for solving (8). In particular, the optimal solution of (11) in the u-space, $u^* = \sum_{\bar{u}\in\mathcal{B}} \bar{u}\theta^*_{\bar{u}}$ where θ^* is its optimal solution in the θ-space, can be shown, under mild conditions, to converge to a solution of (8). Its representative $z^* = b - Au^*$ in the dual space can also be used to keep the size of $\mathcal{B}$ bounded; under appropriate conditions on D_t [13], it is possible to prune down the bundle to any fixed size—downto $\mathcal{B} = \{u^*\}$—without impairing convergence, provided that u^* is inserted to make up for the lost information. A milder way to attain the same result is to keep in $\mathcal{B}$ all the $\bar{u}$ such that $\theta^*_{\bar{u}} > 0$. If c is not concave and/or U is not convex, f is nonetheless a convex function and (5) is equivalent to (and, therefore, bundle methods solve) the "close-convexified version" of (1)

$$(\tilde{\Omega}) \qquad \sup_u \{\ \tilde{c}(u)\ :\ Au = b\ \}$$

where $\tilde{c} = (c + I_U)^{**}$ [33]; when c is linear, for instance, this simply amounts to replacing U with $co\,U$ in (8). These results immediately extend to inequality constraints $Au \leq b$ (yielding sign constraints $x \geq 0$ in the primal); conversely, the generic nonlinear case $A(u) \leq b$ is considerably more complex, even in the convex case [33].

As previously remarked, a modification of the approach is possible when f is a sum-function. This happens e.g. when (8) is a block-structured problem where the constraints $Au = b$ link together blocks of variables that would otherwise be independent, i.e.,

$$(\Omega) \qquad \sup_u \{\ \textstyle\sum_{k\in\mathcal{K}} c^k(u^k)\ :\ \sum_{k\in\mathcal{K}} A^k u^k = b\ ,\ u^k \in U^k \quad k \in \mathcal{K}\ \}\ . \tag{12}$$

It this case, rather than aggregating the individual subgradients z^k—which are provided by the independent solution of each subproblem—into the unique subgradient $z = \sum_{k\in\mathcal{K}} z^k$, one is better off to construct independent models for each component; the corresponding *disaggregated* (primal and dual) master problems

$$(\Pi_{\mathcal{B},\bar{x},t}) \qquad \inf_d \{\ \textstyle\sum_{k\in\mathcal{K}} f^k_{\mathcal{B}}(\bar{x} + d) + D_t(d)\ \} \tag{13}$$

$$(\Delta_{\mathcal{B},\bar{x},t}) \quad \inf_z \{\ \textstyle\sum_{k\in\mathcal{K}} (f^k_{\mathcal{B}})^*(z_h) - \left(\ \sum_{k\in\mathcal{K}} z_k\ \right) \bar{x} + D_t^*\left(-\sum_{k\in\mathcal{K}} z_h\right)\ \} \tag{14}$$

are then solved instead of the aggregated versions. In the Lagrangian case, assuming c linear for simplicity (and disregarding constant terms) (14) is equivalent to the *approximated generalized augmented Lagrangian*

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \sup_u \{\ \textstyle\sum_{k\in\mathcal{K}} (c^k - \bar{x}A^k)u^k - D_t^*\left(\sum_{k\in\mathcal{K}} A^k u^k - b\right)\ :\ u^k \in U^k_{\mathcal{B}} \quad k \in \mathcal{K}\ \} \tag{15}$$

where $U^k_{\mathcal{B}} = co\ \mathcal{B}^k$, that is, to each sub-component U_k of the feasible region (whose cartesian product makes U) to be separately inner-approximated. We will push this approach further, recognizing that for some type of sets U^k inner-approximation is not the most efficient representation. Actually, we will concentrate on sets where there is no need of inner *approximating* anything, because a "compact" representation *exists and is known* that can be directly added to the (dual) master problem.

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

Consider the Lagrangian function f of the following structured optimization problem

$$(\Omega) \qquad \sup_{u_1,u_2} \{\ c_1u_1 + c_2(u_2)\ :\ u_1 \in U^1\ ,\ u_2 \in U^2\ ,\ A_1u_1 + A_2u_2 = b\ \} \qquad (16)$$

where both Lagrangian problems

$$\begin{aligned} f^1(x) &= \sup_{u_1} \{\ (c_1 - xA_1)u_1\ :\ u_1 \in U^1\ \} \\ f^2(x) &= \sup_{u_2} \{\ c_2(u_2) - (xA_2)u_2\ :\ U^2 = \{\ u_2\ :\ G(u_2) \le g\ \}\ \} \end{aligned} \qquad (17)$$

are "easy", but for *different reasons.* In particular, let U^1 be any set for which a "reasonably efficient" (linear) optimization oracle exists, however exotic it may be (e.g. [17]), that can be embedded into the standard "very opaque black box" of Lagrangian optimization. Conversely, assume that $c_2()$ is concave and U^2 is a convex set such that (17) can be solved by any appropriate convex solver (e.g., the convex constraint function $G : \mathbb{R}^n \to \mathbb{R}^m$ is explicitly known and m is "small"). Actually the linearity of c_1 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. Then, at each iteration the bundle approach computes a feasible pair (u_1, u_2) by invoking both, and add them to the respective bundle $\mathcal{B}^1$ and $\mathcal{B}^2$; this results in a dual master problem with the form of (15), where both U^1 (actually, its convex hull) and U^2 are inner approximated by a finite subset of their (extreme) points. This looks very natural, until one examines more in details some relevant case. For instance, in the application of interest here and in others [19], U^2 is the unitary hypercube in the n-dimensional space, and c_2 is linear. So, the *whole* U^2 can be *completely* described by means of $2n$ constraints, while in the standard (disaggregated) bundle approach it is rather iteratively approximated by means of a subset of its 2^n extreme points (cf. §3.2). In light of this example, the standard bundle approach suddenly looks somewhat less natural, or at least less smart: one is blatantly using the "wrong" description of U^2. Fortunately, using the "right" description instead is possible.

The idea is, on the outset, simple: form (13) where only f^1 is approximated by some (e.g., cutting plane) model $f^1_{\mathcal{B}}$, while f^2 is just kept as is

$$(\Pi_{\mathcal{B},\bar{x},t}) \qquad \inf_d \{\ b(\bar{x}+d) + f^1_{\mathcal{B}}(\bar{x}+d) + f^2(\bar{x}+d) + D_t(d)\ \} \quad . \qquad (18)$$

The corresponding stabilized dual master problem is then

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \inf_{z_1,z_2} \Big\{\ (f^1_{\mathcal{B}})^*(z_1) + (f^2)^*(z_2) - \big(z_1 + z_2 + b\big)\bar{x} + D^*_t\big(-b - z_1 - z_2\big)\ \Big\} \quad . \ (19)$$

Note in the above derivation that the Lagrangian function actually has *three* terms: f^1, f^2, and $f^0(x) = bx$. So, one has a z_0 besides z_1 and z_2 in the dual, and an extra term $(f^0)^*(z_0)$ in the objective function: however, the conjugate of the linear function bx is $I_{\{\ b\ \}}$, and this, via $z_0 = b$, gives (19). This passage, that we have purposely not discussed in the general derivation of §2.1, 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. To make (19) implementable one has to choose an appropriate model $f^1_{\mathcal{B}}$, e.g. the cutting plane one. Then, by the very same development as in (10) for the whole of (Ω), one has

$$(f^2)^*(z) = -\sup_{u_2}\left\{\, c_2(u_2) \;:\; z = -A_2u_2 \;,\; G(u_2) \le g \,\right\} \tag{20}$$

under the technical assumptions ensuring no gap between the u_2-problem and its Lagrangian dual. These are very mild, and range from simple compactness of the feasible set, to the Slater constraint qualification, to "regular enough " $G(u_2) \le g$ constraints, such as linear, quadratic or semidefinite [31, Theorem 22]. In practice, any formulation that has a working algebraic dual, such as conic linear programs, satisfies these assumptions. Plugging (20) into (19) for $f^1_{\mathcal{B}} = \hat{f}^1_{\mathcal{B}}$ gives the implementable form

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \sup_{\theta,u_2}\left\{\begin{array}{l} c_1\left(\sum_{\bar{u}_1\in\mathcal{B}}\bar{u}_1\theta_{\bar{u}_1}\right) + c_2(u_2) + \bar{x}z - D^*_t(-z) \\ z = b - A_1\left(\sum_{\bar{u}_1\in\mathcal{B}}\bar{u}_1\theta_{\bar{u}_1}\right) - A_2u_2 \;,\; G(u_2) \le g \;,\; \theta \in \Theta \end{array}\right. \tag{21}$$

with the corresponding "abstract form"

$$(\Omega_{\mathcal{B},\bar{x},t}) \qquad \sup_{z,u_1,u_2}\left\{\begin{array}{l} c_1u_1 + c_2(u_2) + \bar{x}z - D^*_t(-z) \\ z = b - A_1u_1 - A_2u_2 \;,\; u_1 \in U^1_{\mathcal{B}} \;,\; u_2 \in U^2 \end{array}\right. .$$

So, in terms of the original problem the idea is quite straightforward: because U^2 is an "easy" set, one just uses its representation within the master problem, while keeping all the rest of the (approximated generalized augmented) Lagrangian approach unchanged.

It is easy to check from the definition that all the information required by a bundle approach can be obtained by an optimal solution to (21): d^* and $f^1_{\mathcal{B}}(\bar{x}+d^*)$ are the dual optimal multipliers of constraints $z = b - A_1\left(\sum_{\bar{u}_1\in\mathcal{B}}\bar{u}_1\theta_{\bar{u}_1}\right) - A_2u_2$ and $\sum_{\bar{u}_1\in\mathcal{B}}\theta_{\bar{u}_1} = 1$, respectively, $z^*_2 = -A_2u^*_2$, $(f^2)^*(z^*_2) = -c_2(u^*_2)$, and so on. Actually, if U^2 and c_2 are "nice enough" even the *primal* of (21) can be written: if G and c_2 are linear, for instance, one has

$$(\Pi_{\mathcal{B},\bar{x},t}) \qquad \inf_{d,\omega,v}\left\{\begin{array}{l} b(\bar{x}+d) + v + g\omega + D_t(d) \\ v \ge \big(c_1 - (\bar{x}+d)A_1\big)\bar{u}_1 \quad \bar{u}_1 \in \mathcal{B} \\ \omega G = c_2 - (\bar{x}+d)A_2 \;,\; \omega \ge 0 \end{array}\right. \tag{22}$$

where ω are the dual variables of the $Gu_2 \le g$ constraints in (17). Similar results hold whenever c_2 and G allow for closed-form duals, such as those expressible as generic conic programs; note that these typically satisfy the necessary technical assumptions.

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 [13, §5]—and the *real* function f^k cannot but satisfy those requirements. Usually, the crucial 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 still is required, and no 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.

An interesting observation is that one can entirely avoid the oracle for the easy components. This requires that the function value is *only* computed at $x_+ = \bar{x} + d^*$; 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 the (22). This requires doing away with line- or curved-searches; yet this is what happens anyway in most practical implementations, and the convergence theory requires computing $f(x_+)$ at least "frequently enough" [13]. A minor issue with this approach is that at the very first iteration—when (22) 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" [13].

Another relevant feature of bundle methods, aggregation, is not impacted at all from the use of easy components. Simply, one only aggregates on the "difficult" components—e.g., $u_1^* = \sum_{\bar{u}_1 \in \mathcal{B}} \bar{u}_1 \theta^*_{\bar{u}_1}$ in (21)—and prunes their bundles after having inserted the aggregated solution—downto $\mathcal{B} = \{u_1^*\}$ in (21). 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.

2.3 The constrained case

Bundle methods easily cope with constraints, provided the set X is known and (closed) convex; basically, all that is needed is to insert full knowledge about X into (4), i.e., solving

$$(\Pi_{\mathcal{B},\bar{x},t}) \qquad \inf_d \left\{ f_{\mathcal{B}}(\bar{x}+d) + D_t(d) \;:\; (\bar{x}+d) \in X \right\} \tag{23}$$

which is nothing but (4) using the model $f_{X,\mathcal{B}} = f_{\mathcal{B}} + I_X$ of the *essential objective* $f_X = f + I_X$. Its rather abstract dual

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \inf_{z,w} \left\{ f^*_{\mathcal{B}}(z) + \sigma_X(w) - \bar{x}(z+w) + D^*_t(-z-w) \right\}$$

becomes a lot more comfortable e.g. for polyhedral $X = \{ x \in \mathbb{R}^n \;:\; Hx \leq h \}$:

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \inf_{z,\omega} \left\{ f^*_{\mathcal{B}}(z) + \omega h - \bar{x}(z+\omega H) + D^*_t(-z-\omega H) \;:\; \omega \geq 0 \right\}$$

(ω being the dual variables of the optimization subproblem implicit in the definition of σ_X). For $f^1_{\mathcal{B}} = \hat{f}^1_{\mathcal{B}}$ this finally gives a fully implementable dual master problem

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \sup_{\theta,u_2,\omega} \left\{ \begin{array}{l} c_1\left(\sum_{\bar{u}_1\in\mathcal{B}} \bar{u}_1\theta_{\bar{u}_1}\right) + c_2(u_2) + \omega h + \bar{x}z - D^*_t(-z) \\ z = b + \omega H - A_1\left(\sum_{\bar{u}_1\in\mathcal{B}} \bar{u}_1\theta_{\bar{u}_1}\right) - A_2 u_2 \\ G(u_2) \leq g \quad , \quad \theta \in \Theta \quad , \quad \omega \geq 0 \end{array} \right. ; \tag{24}$$

for instance, sign constraints $x \geq 0$, associated to inequality constraints in (Ω), simply give rise to *slack variables* ω in the constraint defining z. The primal of (24), when it is easily written (cf. (22)), simply contains the explicit constraint $Hd \leq h - H\bar{x}$. 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 [13, §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 $+\infty$ 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 evaluate to $+\infty$. However, with $X^k = dom\ f^k$ 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 e.g. in the Lagrangian case 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. [16] 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 [28] 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 (24)—is that the "crucial" optimal aggregated subgradient z^* is given by $z^* = s_0^* + s_X^* + s_1^* + s_2^*$, where $s_0^* = b$, $s_X^* = \omega^* H$, $s_1^* = -A_1 u_1^*$ (with u_1^* obtained by θ^*) and $s_2^* = -A_2 u_2^*$ are (approximated) subgradients of the four components 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 [28], one could first solve a master problem where the constraints $(\bar{x} + d) \in X$ in (23), that is I_X, are replaced by a linearization using s_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 (s_1^* and s_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 (23), which corresponds to one round of a block-Gauss-Seidel approach to (24) where first ω is kept fixed (to the optimal value ω^* 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 [28] 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 (24) with ω and θ fixed is easily solved with specialized approaches due to the structure of U^2 (which is often an "easy" set, cf. e.g. §3.2). 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 [29], so convergence of such an approach could be relatively easy to obtain; this is 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. §3.4).

2.4 Exploiting lower bounds

In practice, it is often the case that global lower bounds are available, either on some of the components ($l^k \leq f^k(x)\ \forall x \in X$), or on the whole objective function ($l \leq f(x)\ \forall x \in X$); 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 requires adding the single item $(0, (f^1)^*(0) = l^1)$ to $\mathcal{B}^1$; with $f^1_{\mathcal{B}} = \hat{f}^1_{\mathcal{B}}$, for instance, this simply results in a new constraint $v \geq l^1$ in (22), and thus in a new variable ρ_1 in (21), with cost l^1, that does not appear in the definition of z, but it does in the constraint $\sum_{\bar{u}_1 \in \mathcal{B}} \theta_{\bar{u}_1} + \rho_1 = 1$. 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 null subgradient belonging to *both* $\mathcal{B}^1$ and $\mathcal{B}^2$ (a unique ρ variable, distinct from the "individual" ρ_1 and ρ_2, participating in both constraints $\sum_{\bar{u}_k \in \mathcal{B}^k} \theta_{\bar{u}_k} = 1$).

Matters are more complex when f^2 is "easy", as one than needs a model of the bounded function $\bar{f}(x) = \max\{\ f(x)\ ,\ l\ \}$, whose conjugate (consult e.g. [23]) is

$$epi\ \bar{f}^* = cl\ co\ \big(\ epi\ f^*\ , epi\ l^*\ \big)\quad , \qquad \text{where} \qquad l^*(z) = I_{\{\ -l\ \}}(z)\quad .$$

Thus, the epigraph of $\bar{f}^*$ can be constructed by taking all points (in the epigraphical space) $(z, f^*(z))$ and computing their convex hull with the point $(0, -l)$; the function value is then the inf over all these possibilities for a fixed z. In a formula,

$$\bar{f}^*(z) = \inf_{z',\rho}\big\{\ \rho f^*(z') - l(1-\rho)\ :\ z = \rho z'\ ,\ \rho \in [0,1]\ \big\}\quad . \tag{25}$$

Note that we are assuming $l \leq \inf f$, thus $f^*(0) \geq -l$: $(0, -l)$ is outside (or at most on the frontier of) *epi* f^*. Hence, the "generic" stabilized dual master problem (6) becomes

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \inf_{z',\rho}\big\{\ \rho f^*(z') - l(1-\rho) - \rho z'\bar{x} + D^*_t(-\rho z')\ :\ \rho \in [0,1]\ \big\}$$

where we have substituted the original variable z with $\rho z'$, giving rise to nasty bilinear terms. Analogously, our disaggregate case with an easy component is

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \inf_{z,z'_1,z'_2,\rho} \left\{ \begin{array}{l} \rho(f^1_{\mathcal{B}})^*(z'_1) + \rho(f^2)^*(z'_2) - l(1-\rho) - z\bar{x} + D^*_t(-z) \\ z = \rho\,(b + z'_1 + z'_2)\ ,\ \rho \in [0,1] \end{array} \right. \tag{26}$$

where we prefer keeping track of the original variable z (not that this spares us with the bilinear term, though). Using the cutting plane model for the first component and the compact representation for the second component one has

$$\begin{array}{rcl} \rho(f^1_{\mathcal{B}})^*(z'_1) & = & -\sup_\theta\big\{\ \rho c_1 \sum_{\bar{u}_1 \in \mathcal{B}} \bar{u}_1\theta_{\bar{u}_1}\ :\ \sum_{\bar{u}_1 \in \mathcal{B}} \bar{u}_1\theta_{\bar{u}_1} = z'_1\ ,\ \theta \in \Theta\ \big\} \\ \rho(f^2)^*(z'_2) & = & -\sup_{u_2}\big\{\ \rho c_2(u_2)\ :\ z'_2 = -A_2u_2\ ,\ G(u_2) \leq g\ \big\} \end{array}$$

which in turn yields

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \sup_{\rho,z,\theta,u_2} \left\{ \begin{array}{l} \rho c_1 \sum_{\bar{u}_1 \in \mathcal{B}} \bar{u}_1\theta_{\bar{u}_1} + \rho c_2(u_2) - l(1-\rho) + z\bar{x} - D^*_t(-z) \\ z = \rho\,\big(b - A_1 \sum_{\bar{u}_1 \in \mathcal{B}} \bar{u}_1\theta_{\bar{u}_1} - A_2u_2\big)\ ,\ G(u_2) \leq g\ ,\ \theta \in \Theta\ ,\ \rho \in [0,1] \end{array} \right. .$$

This can be brought to a fully implementable master problem if c_2 and G are linear and U_2 is compact. In fact, in this case the bilinear terms can be eliminated by substituting z with its definition and the change of variables $\theta' = \rho\theta$ and $u'_2 = \rho u_2$, which finally gives

$$(\Delta_{\mathcal{B},\bar{x},t}) \qquad \sup_{\rho,z,\theta',u_2'} \left\{ \begin{array}{l} c_1 \sum_{\bar{u}_1 \in \mathcal{B}} \bar{u}_1 \theta'_{\bar{u}_1} + c_2 u'_2 - l(1-\rho) + z\bar{x} - D^*_t(-z) \\ z = \rho b - A_1 \sum_{\bar{u}_1 \in \mathcal{B}} \bar{u}_1 \theta'_{\bar{u}_1} - A_2 u'_2 \quad , \quad \rho = \sum_{\bar{u}_1 \in \mathcal{B}} \theta'_{\bar{u}_1} \\ G u_2 \leq \rho g \ \ , \ \ \theta' \geq 0 \ \ , \ \ \rho \in [0,1] \end{array} \right. .$$

Note that the original variables θ (hence, u_1) and u_2 must be obtained by scaling them by $1/\rho$ after the problem has been solved, except if $\rho = 0$ when $\{ \ u_2 \ : \ G(u_2) \leq 0 \ \} \ = \ \{ \ 0 \ \}$ by compactness of U_2. The corresponding primal is

$$(\Pi_{\mathcal{B},\bar{x},t}) \qquad -\, l + \inf_{d,v,v_1} \left\{ \begin{array}{lll} v + D_t(d) & & \\ v_1 \geq \big(c_1 - (\bar{x}+d)A_1\big)\bar{u}_1 & & \bar{u}_1 \in \mathcal{B} \\ v \geq b(\bar{x}+d) + v_1 + gy & , & v \geq l \\ yG = c_2 - (\bar{x}+d)A_2 & , & y \geq 0 \end{array} \right.$$

Incorporating individual lower bounds $f^1_{\mathcal{B}} \geq l^1$ in the above development is straightforward.

The above reformulations are no longer possible when c_2 and/or G are nonlinear, making incorporation of a global lower bound more difficult. Analogously, while the treatment immediately extends to any model $f^1_{\mathcal{B}}$ such that $(f^1_{\mathcal{B}})^*$ has a polyhedral representation, it does not extend easily to non-polyhedral models; a significant exception is that when the model can be expressed by mean of a *conic program* (e.g. [1]).

3 Application: the Fixed-Charge Multicommodity Min-Cost Flow problem

3.1 Formulation

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 $b^k = [b^k_i]_{i \in N}$ indicating the net amount of flow required by the node, so that 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, e.g. in telecommunications, 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., $b^k_i = -d^k$ if $i = s_k$, $b^k_i = d^k$ if $i = t_k$, and $b^k_i = 0$ 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 an aggregated (mutual) arc capacity $u_{ij} > 0$. Also, individual arc capacities u^k_{ij} 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 pair case one may set $u^k_{ij} = d^k$, which is useful if $d^k \ll u_{ij}$. Furthermore, a routing cost c^k_{ij} has to be paid for each unit of commodity k moving through (i,j). The problem consists in minimizing the sum of all costs, while satisfying demand requirements and capacity constraints.

Defining *arc flow variables* w^k_{ij}, which represent the amount of the flow of commodity k 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 *arc flow formulation* of the problem is

$$\min \textstyle\sum_{k\in K}\sum_{(i,j)\in A} c^k_{ij} w^k_{ij} + \sum_{(i,j)\in A} f_{ij} y_{ij} \tag{27}$$

$$\textstyle\sum_{(j,i)\in A} w^k_{ji} - \sum_{(i,j)\in A} w^k_{ij} = b^k_i \qquad i \in N \;,\; k \in K \tag{28}$$

$$\textstyle\sum_{k\in K} w^k_{ij} \leq u_{ij} y_{ij} \qquad (i,j) \in A \tag{29}$$

$$0 \leq w^k_{ij} \leq u^k_{ij} \qquad (i,j) \in A \;,\; k \in K \tag{30}$$

$$y_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{31}$$

Model (27)–(30) is known as the *weak formulation* because the lower bound produced by the corresponding continuous relaxation is usually weak. The *strong formulation* can be obtained by replacing (30) with

$$0 \leq w^k_{ij} \leq u^k_{ij} y_{ij} \qquad (i,j) \in A \;,\; k \in K \tag{32}$$

which is obviously valid, and useful if $u^k_{ij} < u_{ij}$.

3.2 Decomposition approaches

FC-MMCF lends itself nicely to the application of Lagrangian techniques for the computation of lower bounds, that can then be effectively incorporated into either heuristic [22, 9, 21] or exact [25, 30] approaches. Indeed, the multicommodity flow structure has always been a favorite target for Lagrangian approaches [8, 7, 15, 16, 20, 26, 32]. Here we consider the Lagrangian relaxation of constraints (29) with multipliers $\alpha = [\ \alpha_{ij}\ ]_{(i,j)\in A} \geq 0$, which gives the (concave) Lagrangian function

$$f(\alpha) = \left\{ \begin{array}{ll} \min & \sum_{k\in K}\sum_{(i,j)\in A} (c^k_{ij} + \alpha_{ij}) w^k_{ij} + \sum_{(i,j)\in A} (f_{ij} - \alpha_{ij} u_{ij}) y_{ij} \\ & \qquad\qquad (28) \;\;,\;\; (30) \;\;,\;\; (31) \end{array} \right.$$

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

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

i.e., the classical (single-commodity) *min-cost flow* structure, for which plenty of efficient solution algorithms exist [18, 20]. Also, linear optimization on the set Y given by the simple constraints (31) is very easy; actually, $Y = \bigotimes_{(i,j)\in A} Y^{ij}$ itself is separable into 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; this is also true for the strong formulation, where constraints (32) are also relaxed in Lagrangian fashion with multipliers $\beta = [\ \beta^k_{ij}\ ]_{(i,j)\in A\,,\,k\in K} \geq 0$, yielding

$$f(\alpha, \beta) = \left\{ \begin{array}{ll} \min & \sum_{(i,j)\in A} \Big(\sum_{k\in K} (c^k_{ij} + \alpha_{ij} + \beta^k_{ij}) w^k_{ij} + (f_{ij} - \alpha_{ij} u_{ij} - \sum_{k\in K} \beta^k_{ij} u^k_{ij}) y_{ij} \Big) \\ & \qquad\qquad (28) \;\;,\;\; (30) \;\;,\;\; (31) \end{array} \right.$$

(note that the constraints (30), redundant when (32) are present, are used to tighten up the relaxation).

For both functions, several possibilities exist when solving the corresponding minimization 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, where the separability in f is totally ignored and only one subgradient of the function is produced;
- a *partly disaggregated with easy y* (PDE) version, where the W and Y components are treated as *two* distinct functions, and with the Y component treated as an "easy" one;
- a *disaggregated with difficult y* (DD) version, where the W component is decomposed into its $|K|$ distinct functions, while the Y component is treated as *one* other function;
- a *disaggregated with easy y* (DE) version, which is the same as the above but with the Y component treated as an "easy" one.

It is instructive to present the master problem of the DE formulation under some simplyfing assumptions that may render it more familiar to some readers. Let us assume that commodities are origin-destination pairs; in this case, the relevant extreme points of each set W^k correspond to elements $p \in \mathcal{P}^k$, the set of all s_k–t_k paths. Hence, defining *path flow variables* f_p for each $p \in \mathcal{P} = \cup_{k \in K} \mathcal{P}^k$, one can consider the (weak) *path flow formulation*

$$\min \textstyle\sum_{p \in \mathcal{P}} c_p f_p + \sum_{(i,j) \in A} f_{ij} y_{ij} \tag{33}$$

$$\textstyle\sum_{p \in \mathcal{P} \,:\, (i,j) \in p} f_p \leq u_{ij} y_{ij} \qquad (i,j) \in A \tag{34}$$

$$\textstyle\sum_{p \in \mathcal{P}^k} f_p = d^k \qquad k \in K \tag{35}$$

$$f_p \geq 0 \qquad p \in \mathcal{P} \tag{36}$$

$$y_{ij} \in [0,1] \qquad (i,j) \in A \tag{37}$$

where c_p is the cost of the path p (sum of the costs of its arcs); the equivalent of constraints (32) could be easily added to obtain the corresponding strong version, which we avoid only for the sake of notational simplicity. It is easy to realize that, modulo a scaling of the variables, this is precisely what the master problem of DE would look like if: a) one had generated all possible extreme points of each W^k already, and b) one would be using the non-stabilized the cutting-plane method, i.e., choose $D_t(\cdot) = 0$ which corresponds to $D_t^*(\cdot)$ being "the constraint $z = 0$". Said the other way around, DE corresponds to: a) doing column generation on the exponential-size reformulation (33)–(37), after b) having replaced the linking constraints (34) by a slackened version, appropriately penalizing the slacks z in the objective function (cf. (21)). This can be defined a *Stabilized Partial Dantzig-Wolfe Decomposition*, in that only a subset of the feasible region is reformulated by means of its extreme points and iteratively inner approximated. Now, DD would be obtained from that by the substitution

$$y = \textstyle\sum_{s \in \mathcal{S}} \bar{y}^s \theta_s \qquad \text{with constraints} \qquad \sum_{s \in \mathcal{S}} \theta_s = 1 \quad , \quad \theta_s \geq 0 \quad s \in \mathcal{S}$$

where $\mathcal{S}$ is the set of all extreme points $\bar{y} \in \{0,1\}^{|A|}$ of Y, i.e., the $2^{|A|}$ vertices of the unitary hypercube. Looked at in this way, it is not much surprising that, as shown in §3.4, the versions with "easy" components are much better than those using the standard approach; the latter use a blatantly "wrong" representation of Y, arguably the worst possible one. Yet, this approach—as ungainly as it looks when seen from the primal viewpoint—does not seem to have ever been questioned before.

One important observation is that a further possibility exists: a *fully disaggregated* (FD) version where the W component is decomposed into its $|K|$ distinct functions, and the Y component is decomposed into its $|A|$ distinct functions. However, this is basically equivalent to DE. In fact, for each $(i,j) \in A$ there are only two extreme points $\bar{y}_{ij} = 0$ and $\bar{y}_{ij} = 1$.

If both are inserted into the bundle $\mathcal{B}^{ij}$ for that component, with multipliers $\theta^{ij,0}$ and $\theta^{ij,1}$, respectively, it is easy to verify that one is basically substituting the y_{ij} variable with

$$y_{ij} = 0 \cdot \theta^{ij,0} + 1 \cdot \theta^{ij,1} \ , \ \theta^{ij,0} + \theta^{ij,1} = 1 \ , \ \theta^{ij,0} \geq 0 \ , \ \theta^{ij,1} \geq 0$$

which is of course is nothing but a convoluted way to write (37). Therefore, in the following we will not test the FD variant.

The last remark may have mislead the reader into thinking that the whole "easy component" idea is then just a special case of, or a minor improvement upon, the standard approach of exploiting the existing sum-structure of the function by disaggregating the master problem accordingly. This clearly isn't so. For instance, consider the slight variant of FC-MMCF where one requires that at most a given number h of arcs are "opened". In the model, this corresponds to the simple extra constraint

$$\textstyle\sum_{(i,j)\in A} \ y_{ij} \leq h \tag{38}$$

whose immediate effect is to kill separability of the Y set. Thus, in such a variant the maximum possible degree of disaggregation is $|K| + 1$, as all the y variables must belong to the same subproblem, and the FD variant is no longer viable. However, the DE variant is still perfectly possible, with basically the single constraint (38) added to the master problem, because the corresponding set Y still has the integrality property. Thus, while being equivalent to the FD approach for FC-MMCF, the "easy" component idea is clearly more general.

3.3 Stabilizing functions

The generalized bundle algorithm is largely independent on the choice of the stabilizing term D_t, provided that the a few weak conditions are satisfied. Indeed, by looking at (21)/(24), one notices that the choice of D_t only impacts on the $D^*_t(-z)$ 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; all are separable, that is, $D_t(d) = \sum_{i=1}^n \Psi_t(d_i)$ and $D^*_t(z) = \sum_{i=1}^n \Psi^*_t(z_i)$, where $\Psi_t \ : \ \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ are convex. Notable examples are:

- *Trust Region/BoxStep*: This uses $\Psi_t = I_{[-t,t]}$, that is, it establishes a trust region of "radius t" around the current point. From the dual viewpoint, this corresponds to $\Psi^*_t = t|\cdot|$, i.e., to a linear stabilization.
- *Proximal stabilization*: This uses $\Psi_t = \frac{1}{2t}(\cdot)^2$, $\Psi^*_t = \frac{1}{2}t(\cdot)^2$, hence both the primal and dual master problems are separable convex quadratic problems.
- *Linear-quadratic penalty*: This is a modification of the boxstep method where

$$\Psi^*_{t,\varepsilon}(z) = t \begin{cases} z^2/\varepsilon & \text{if } -\varepsilon \leq s \leq \varepsilon \\ |z| & \text{otherwise} \end{cases} \qquad \Psi_{t,\varepsilon}(d) = \begin{cases} \frac{\varepsilon}{4t}d^2 & \text{if } -t \leq d \leq t \\ +\infty & \text{otherwise} \end{cases}$$

  and therefore nonsmoothness at zero of D^*_t is avoided.

Actually, the treatment can be extended to the case when the stabilizing term depends on multiple parameters instead of just one. For instance, the 5-*piecewise linear penalty function*

$$\Psi_t(d) = \begin{cases} +\infty & \text{if} \quad d \leq -(\Gamma^- + \Delta^-) \\ -\varepsilon^-(d - \Delta^-) & \text{if} \quad -(\Gamma^- + \Delta^-) \leq d \leq -\Delta^- \\ 0 & \text{if} \quad -\Delta^- \leq d \leq \Delta^+ \\ +\varepsilon^+(d - \Delta^+) & \text{if} \quad \Delta^+ \leq d \leq (\Delta^+ + \Gamma^+) \\ +\infty & \text{if} \quad (\Delta^+ + \Gamma^+) \leq d \end{cases} \quad , \tag{39}$$

whose corresponding 4-piecewise primal penalty is

$$\Psi_t^*(z) = \begin{cases} -(\Gamma^- + \Delta^-)z - \Gamma^-\varepsilon^- & \text{if} \quad -(\zeta^- + \varepsilon^-) \leq z \leq -\varepsilon^- \\ -\Delta^- z & \text{if} \quad -\varepsilon^- \leq z \leq 0 \\ +\Delta^+ z & \text{if} \quad 0 \leq z \leq \varepsilon^+ \\ +(\Gamma^+ + \Delta^+)z + \Gamma^+\varepsilon^+ & \text{if} \quad \varepsilon^+ \leq z \leq (\zeta^+ + \varepsilon^+) \end{cases} . \tag{40}$$

In this case, $t = [\zeta^\pm, \varepsilon^\pm, \Gamma^\pm, \Delta^\pm]$ is a vector of parameters. These and similar piecewise-linear stabilizing functions have been tested (for instance, in [4]), showing that appropriately setting their parameters may lead to faster convergence of stabilized algorithms (in that case, applied to column generation). All these cases can be easily implemented within the same framework. Even the "complicated" (39)/(40) just corresponds to defining $z = z_2^- + z_1^- - z_1^+ - z_2^+$, with

$$\zeta^+ \geq z_2^+ \geq 0 \qquad \varepsilon^+ \geq z_1^+ \geq 0 \qquad \varepsilon^- \geq z_1^- \geq 0 \qquad \zeta^- \geq z_2^- \geq 0$$

in the primal master problem, and objective function

$$(\bar{y} - \Delta^- - \Gamma^-)z_2^- + (\bar{y} - \Delta^-)z_1^- - (\bar{y} + \Delta^+)z_1^+ - (\bar{y} + \Delta^+ + \Gamma^+)z_2^+ \ .$$

Hence, the primal master problem is still a linear program with the same number of constraints and $4m$ new variables. For our experiments we only used the simpler choices, i.e., trust region and proximal stabilization.

3.4 Computational results

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 [5, 8, 19]. 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 [3, 10], and the bundle method. The latter in particular is generalized, in the sense that the solution of the master problem is demanded 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 tested both the specialized quadratic solver described in [12] (for $\Psi_t = \frac{1}{2t}(\cdot)^2$) and the use of general-purpose LP solvers (for $\Psi_t = I_{[-t,t]}$). 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. [20]). All the algorithms have been coded in `C++`, compiled with GNU `g++ 4.4.5` (with `-O3` optimization option) and ran 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 [7], 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. [8]), 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 1. 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 (32). 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	$\lvert N\rvert$	$\lvert A\rvert$	$\lvert K\rvert$
1	20	300	100
2	20	300	200
3	20	300	400
4	20	300	800
5	30	600	100
6	30	600	200
7	30	600	400
8	30	600	800
9	50	1200	100
10	50	1200	200
11	50	1200	400
12	50	1200	800

Table 1: Description of the instances

3.5 Results for the weak formulation

The results are presented in Table 2 for the different approaches discussed in §3.2. In particular, the columns "FA-2" report results for the FA approach with $\Psi_t = \frac{1}{2t}(\cdot)^2$, where the master problem is solved with the specialized quadratic solver of [12], while columns "FA-1" report results for the same approach with $\Psi_t = I_{[-t,t]}$, 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; 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 these 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.

Comparing FA-2 with FA-1, it is clear that the proximal stabilization is in general largely preferable to the trust-region one. This has been reported several times over in different applications [6, 4, 16], and it just proves true once again here. Part of the result is due to the (still) more effective quadratic Master Problem solver of [12] w.r.t. general-purpose LP technology, as testified by the lower cost per iteration; however, the largest part of the

DE			PDE				DD				FA-1				FA-2			
time	f	it	time	f	it	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

improvement comes from faster convergence. Disaggregating X 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. However, "disaggregating Y"—that is, considering Y as an easy component—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 X is disaggregated and Y is treated as an easy component; this ends up being in the region of two order 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 on all the instances 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, 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 [8], even at the lower accuracy setting FA-2 is not competitive with the best that `Cplex` offers (the primal network algorithm, which somewhat surprisingly is *not* automatically chosen, cf. the column "auto"); 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 [8] 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 (27)–(31), 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. Furthermore, requiring the very-high accuracy to DA 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.

3.6 Results for the strong formulation

Due to the huge number of constraints (32), 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 rather 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 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, in `Cplex` violation of lazy constraints is only checked 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 solution. A similar approach can be used for the decomposition approach, and it has already shown to be very effective [15, 19] 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 rows 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 true, as shown in Table 5 again only for the four groups of smaller instances (things only get worse as size increase). 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, is fixed to 1000, 3000, 9000 and 27000 seconds when the number of commodities is 100, 200, 400 and 800, respectively.

DE			PDE			DD			FA-1			FA-2		
time	it	gap	time	it	gap	time	it	gap	time	it	gap	time	it	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, most decomposition approaches do not even come close to the required `1e-6` precision even allowing for such long computing times. 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 solution 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 (32). 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 middle precisions to improve the readability of the Table; the trends closely follows these for the two extreme precisions.

The picture painted by Table 6, albeit not spectacular 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 DA are:

`1e-6`					`1e-8`			`1e-10`			`1e-12`			
time	f	add	it	gap	time	it	gap	time	it	gap	time	f	add	it
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 maximum size of the bundle is set to $50 \cdot |K|$, and subgradients are only removed if their multiplier θ is zero for 40 consecutive iterations;
- constraints 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 [12] 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 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 and as much as 10% of the total running time in some cases. However, the time for solving the Lagrangian function is itself 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 is zero for 20 (as opposed to 40) consecutive iterations, and columns "Sep = 10" are relative to performing separation of constraints (32) every 10 iterations (as opposed to every iteration).

opt				$20 \cdot \lvert K \rvert$				Rem = 20				Sep = 10			
time	add	it	gap	time	add	it	gap	time	add	it	gap	time	add	it	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 DA

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.

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 FA-V, i.e., the fully disaggregated model solved with the Volume algorithm [3]; subgradient-type approaches have often been found competitive with bundle ones for the approximate solution of difficult large-scale problems [8, 19]. 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 [16]; unfortunately, this was not one of these cases.

`Cplex`				DE		FA-2					FA-V				
primal	dual	net.	barr.	`1e-6`	`1e-12`	time	f	add	it	gap	time	f	add	it	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 [8], 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 not implemented (except in a very primitive fashion) in [8]. The Volume algorithm used here is remarkably more "robust" than the subgradient algorithms employed in [8]—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 likely 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, neither FA-V nor FA-2 are competitive with `Cplex`: the former is faster but obtains unacceptably coarse bounds, the latter is slower and the quality of the bounds is lower as well. Conversely, the DE bundle method obtains accurate—even extremely so—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. Appropriate *structure* of the master problem, under the form of the compact representation (as opposed to, perhaps, "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 the application considered in this paper goes.

4 Conclusions

Exploiting structure of the different components of the function in order to improve the performances is a strong recent trend in the research about bundle methods for nondifferentiable optimization [16, 28, 29, 32], 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. One possibility comes from another staple of Lagrangian optimization: the hydro-thermal Unit Commitment problem in electrical power production [2, 5]. There, while thermal units require complex combinatorial oracles [17], hydro units—at least with some widely accepted simplifications—are just small-scale continuous flow problems, and therefore amenable to the "easy components" treatment.

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 [4]) 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 [12] is still effective nowadays is comforting in this respect, showing just how powerful appropriate structure exploitation can be. Yet, for the approach proposed in this paper one would require an oxymoron-sounding "generic-specialized" solver that on one hand exploits the standard structure of the master problem as [12], but on the other hand be able to deal with as many structures of the easy components as possible. The Alternating Linearizaton approach (cf. §2.3) may be a direction to explore, but the task may call for the development of some entirely new solution methodologies for structured quadratic programming.

References

[1] A. Astorino, A. Frangioni, M. Gaudioso, and E. Gorgone. Piecewise Quadratic Approximations in Convex Numerical Optimization. Technical Report 2/10, DEIS, Università della Calabria, 2010.

[2] 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.

[3] L. Bahiense, N. Maculan, and C. Sagastizábal. The volume algorithm revisited: relation with bundle methods. *Mathematical Programming*, 94(1):41–70, 2002.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] G. d'Antonio and A. Frangioni. Convergence Analysis of Deflected Conditional Approximate Subgradient Methods. *SIAM Journal on Optimization*, 20(1):357–386, 2009.

[11] 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.

[12] A. Frangioni. Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Computers & Operations Research*, 21:1099–1118, 1996.

[13] A. Frangioni. Generalized Bundle Methods. *SIAM Journal on Optimization*, 13(1):117–156, 2002.

[14] A. Frangioni. About Lagrangian Methods in Integer Optimization. *Annals of Operations Research*, 139:163–193, 2005.

[15] 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.

[16] A. Frangioni and B. Gendron. A Stabilized Structured Dantzig-Wolfe Decomposition Method. Technical Report CIRRELT-2010-02, CIRRELT, 2010.

[17] A. Frangioni and C. Gentile. Solving Nonlinear Single-Unit Commitment Problems with Ramping Constraints. *Operations Research*, 54(4):767 – 775, 2006.

[18] 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.

[19] A. Frangioni, A. Lodi, and G. Rinaldi. New approaches for optimizing over the semimetric polytope. *Mathematical Programming*, 104(2-3):375–388, 2005.

[20] 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.

[21] I. Ghamlouche, T.G. Crainic, and M. Gendreau. Path Relinking, Cycle-Based Neighbourhoods and Capacitated Multicommodity Network Design. *Annals of Operations Research*, 131(1-4):109–133, 2004.

[22] I. Ghamlouche, T.G. Crainic, and B. Gendron. Cycle-Based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design. *Operations Research*, 51(4):655–667, 2003.

[23] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I—Fundamentals*, volume 306 of *Grundlehren Math. Wiss.* Springer-Verlag, New York, 1993.

[24] 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.

[25] 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.

[26] 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.

[27] K. Kiwiel and C. Lemaréchal. An inexact bundle variant suited to column generation. *Mathematical Programming*, 118:177–206, 2009.

[28] K.C. Kiwiel. A Proximal-Projection Bundle Method for Lagrangian Relaxation, Including Demidefinite Programming. *SIAM Journal on Optimization*, 17(4):1015–1034, 2006.

[29] K.C. Kiwiel. An Alternating Linearization Bundle Method for Convex Optimization and Nonlinear Multicommodity Flow Problems. *Mathematical Programming*, to appear, 2011.

[30] 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.

[31] C. Lemaréchal. Lagrangian Relaxation. In M. Jünger and D. Naddef, editors, *Computational Combinatorial Optimization*, pages 115–160. Springer-Verlag, Heidelberg, 2001.

[32] 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.

[33] C. Lemaréchal and A. Renaud. A geometric study of duality gaps, with applications. *Mathematical Programming*, 90:399–427, 2001.

[34] A. Ouorou. A proximal cutting plane method using Chebychev center for nonsmooth convex optimization. *Mathematical Programming*, 119(2):239–271, 2009.