

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/316770538>

On the computational efficiency of subgradient methods: a case study with Lagrangian bounds

Article in *Mathematical Programming Computation* · May 2017

DOI: 10.1007/s12532-017-0120-7

CITATIONS

2

READS

64

3 authors:



Frangioni Antonio

Università di Pisa

107 PUBLICATIONS 2,134 CITATIONS

[SEE PROFILE](#)



Bernard Gendron

Université de Montréal

98 PUBLICATIONS 2,093 CITATIONS

[SEE PROFILE](#)



Enrico Gorgone

Università degli studi di Cagliari

16 PUBLICATIONS 89 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Delay-constrained Routing [View project](#)



Modeling and solving a logging camp location problem [View project](#)

All content following this page was uploaded by [Frangioni Antonio](#) on 24 May 2017.

The user has requested enhancement of the downloaded file.

1 **On the computational efficiency of subgradient methods:**
2 **a case study with Lagrangian bounds**

3 **Antonio Frangioni · Bernard Gendron ·**
4 **Enrico Gorgone**

5
6 the date of receipt and acceptance should be inserted later

7 **Abstract** Subgradient methods (SM) have long been the preferred way to
8 solve the large-scale Nondifferentiable Optimization problems arising from the
9 solution of Lagrangian Duals (LD) of Integer Programs (IP). Although other
10 methods can have better convergence rate in practice, SM have certain advan-
11 tages that may make them competitive under the right conditions. Further-
12 more, SM have significantly progressed in recent years, and new versions have
13 been proposed with better theoretical and practical performances in some ap-
14 plications. We computationally evaluate a large class of SM in order to assess
15 if these improvements carry over to the IP setting. For this we build a unified
16 scheme that covers many of the SM proposed in the literature, comprised some
17 often overlooked features like projection and dynamic generation of variables.
18 We fine-tune the many algorithmic parameters of the resulting large class of
19 SM, and we test them on two different Lagrangian duals of the Fixed-Charge
20 Multicommodity Capacitated Network Design problem, in order to assess the
21 impact of the characteristics of the problem on the optimal algorithmic choices.
22 Our results show that, if extensive tuning is performed, SM can be competitive
23 with more sophisticated approaches when the tolerance required for solution
24 is not too tight, which is the case when solving LDs of IPs.

25 **Keywords** Subgradient methods, Nondifferentiable Optimization, Compu-
26 tational analysis, Lagrangian relaxation, Multicommodity Network Design

Antonio Frangioni
Dipartimento di Informatica, Università di Pisa, E-mail: frangio@di.unipi.it

Bernard Gendron
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le
Transport (CIRRELT), and Department of Computer Science and Operations Research,
Université de Montréal, E-mail: Bernard.Gendron@cirrelt.ca

Enrico Gorgone
Dipartimento di Matematica ed Informatica, Università di Cagliari, E-mail: egor-
gone@unica.it, and Indian Institute of Management Bangalore (IIMB), E-mail: en-
rico.gorgone@iimb.ernet.in

27 **Mathematics Subject Classification (2000)** 90C06 · 90C25

28 1 Introduction

29 The aim of this paper is to computationally evaluate a large family of ap-
30 proaches for the solution of problems of the form

$$f_* = \min \{ f(\lambda) = \sum_{k \in \mathcal{K}} f^k(\lambda) : \lambda \in \Lambda \} \quad (1)$$

31 where \mathcal{K} is a finite index set, $\Lambda \subseteq \mathbb{R}^n$ is closed, convex and “easy” in the
32 sense that projection upon Λ is inexpensive, and $f^k : \mathbb{R}^n \rightarrow \mathbb{R}$ are proper con-
33 vex functions. The generalized gradient descent method, a.k.a. the *Subgradient*
34 *Method* (SM), is the extension of the gradient method for smooth optimiza-
35 tion introduced in the 60s [61] that solves (1) under very mild conditions.
36 In particular, each of the functions f^k need only be known through a “black
37 box” that, given $\lambda \in \Lambda$, returns the function value $f^k(\lambda)$ and a subgradient
38 $g^k \in \partial f^k(\lambda)$. Then, after computing $f(\lambda)$ according to (1), and similarly for
39 $\sum_{k \in \mathcal{K}} g^k = g \in \partial f(\lambda)$, the algorithm employs the simple recurrence formula

$$\tilde{\lambda}_{i+1} = \lambda_i - \nu_i g_i \quad , \quad \lambda_{i+1} = P_\Lambda(\tilde{\lambda}_{i+1}), \quad (2)$$

40 where P denotes the orthogonal projection on Λ . Only very simple rules are
41 required to the *stepsize* $\nu_i \in \mathbb{R}_+$ to ensure that the sequence $\{f_i = f(\lambda_i)\}$
42 asymptotically solves (1), i.e., $\liminf_{i \rightarrow \infty} f_i = f_\infty = f_*$. Under mild additional
43 assumptions, cluster points of $\{\lambda_i\}$ also are optimal solutions to (1).

44 SM require $\Theta(1/\epsilon^2)$ iterations to solve (1) up to *absolute* error ϵ , which
45 means that they are not practical for attaining any more than a modest ac-
46 curacy. Yet, that is also the *best* possible worst-case complexity for the min-
47 imization of a generic nondifferentiable function only known via a black box
48 [51]. Besides, the complexity is independent of the size n of the problem.
49 Therefore, SM may be promising for very-large-scale problems where a high
50 accuracy is not necessary, whereas a short running time is a primary concern.
51 This happens to be often the case when f is the *Lagrangian function* of a hard
52 optimization problem, say a *block-structured* Integer Program (IP)

$$\max \{ \sum_{k \in \mathcal{K}} c^k u^k : \sum_{k \in \mathcal{K}} A^k u^k = b \quad , \quad u^k \in U^k \quad k \in \mathcal{K} \} \quad , \quad (3)$$

53 where one relaxes, in a Lagrangian fashion, the *complicating constraints* that
54 link together blocks of variables that would otherwise be independent, yielding

$$f(\lambda) = \lambda b + \sum_{k \in \mathcal{K}} (f^k(\lambda) = \max \{ (c^k - \lambda A^k) u^k : u^k \in U^k \}) \quad . \quad (4)$$

55 Often the sets U^k are “hard”, say encompassing integrality restrictions, so that
56 (3) is a “hard” problem. Thus, (4) is less hard than (3) if only because it de-
57 composes into smaller independent subproblems. In some cases (4) is simpler
58 even if $|\mathcal{K}| = 1$ since U^1 has a specific structure that can be algorithmically
59 exploited; sometimes, as in §3.1, both effects apply. Therefore, to simplify the

notation we will write cu , $Au = b$ and U respectively for the objective function, linking constraints and feasible region in (3)/(4) when the sum-function structure is better ignored. We also remark that there is a slight (and intended) inconsistency between (4) and (1), in that the former actually has $|\mathcal{K}| + 1$ functions counting the linear one λb ; we will ignore this detail (say, assume $b = 0$) up until it becomes relevant.

The *Lagrangian relaxation* (4) of the IP (3), although not the only application of SM, has been one of the main factors motivating the interest in this class of algorithms. After the seminal [36], the use of *Lagrangian Duals* (LD) [32] has been a staple of integer optimization for over two decades, during which “Lagrangian relaxation” has invariably been a synonym of “solving a LD by a SM.” In fact, some of the improvements on the SM originate from the IP community, such as the *deflection techniques* introduced in [14] to face the “zig-zagging” behaviour whereby $g_{i+1} \approx -g_i$, so that two “reasonably long” steps combined make an “unfeasibly short” one. This leads to replace (2) with

$$\tilde{\lambda}_{i+1} = \lambda_i - \nu_i d_i \quad (5)$$

where the *direction* d_i is obtained by some linear combination of the *current subgradient* g_i and the *previous direction* d_{i-1} . In the constrained case, the fact that d_i is chosen without regard of the feasible set Λ also independently causes the zig-zagging phenomenon, unless *conditional subgradient* techniques [45] are employed whereby d_i is first *projected* on the tangent cone of Λ at λ_i (it is somewhat surprising that the combination of deflection and projection has not been analyzed until [21]). Again, IP has been the main motivation for their development: inequality constraints $Au \leq b$ in (3) give $\Lambda = \mathbb{R}_+^n$. Also, stepsize rules have been developed specifically for integer optimization [6, 31].

The appeal of SM has started to decrease during the early 90s, for different reasons. On one hand, the success of polyhedral techniques has meant that Branch&Cut (B&C) approaches based on standard Linear Programming (LP) techniques have rapidly become the method of choice for the solution of IPs. On the other hand, Cutting-Plane (CP) methods for solving (1) had been known for almost as long as SM [40], and variants have been developed over the years that have been proven to be superior to SM in many circumstances. In particular, both Bundle methods [37, 47, 64] and center-based methods [22] (the latter often [33], but not always [57], based on interior-point techniques) *stabilize* the original CP, most of the time resulting in the best performances [10, 13, 19, 30]. Yet, the computational advantage of these methods upon SM is mostly seen “at the tail” of the process, where SM convergence remains slow whereas other methods can (if properly set) rapidly accrue “the right set of information to stop” [29, 30]. In earlier stages the behaviour is more similar, despite the fact that the other methods hoard much more information than SM do [13]. This implies a higher cost per iteration due to the solution of the *Master Problem* (MP), that can account for a large part of the total computational time [27, 29], thereby possibly negating the advantage due to faster convergence. Although the cost of the MP can be decreased, e.g. by developing

specialized methods [24] or by reformulating it so that it can be more effectively solved by standard ones [8], the SM is inherently less expensive. The MP cost is particularly hurtful when solving the subproblems in parallel, since then the MP is the sequential bottleneck that limits the parallel speedup [15].

Furthermore, research on SM continued to bring improvements. One was the re-discovery [5,6,46] of what should have been a well-known property [2], i.e., that SM can be endowed with ways to produce (approximately) optimal solutions to the *convexified relaxation* [48]. This puts them on par with Bundle and center-based methods, that have always been well-known for being able to produce primal solutions [23,26] as a by-product of dual optimization. Also, *incremental SM* have been proposed [41,50,56] which allow to exploit the block-separable structure of (4) to potentially speed-up computations, something that—albeit with a very different set of trade-offs—Bundle methods were already well-known to be able to do [4,10,22,29,39]. Finally, *fast SM* have been proposed, starting with [52,53], which try to exploit structure in f to close the gap with *fast gradient methods* [51], that have substantially better convergence rates than SM but require differentiability of f . Applied to our setting these would require to solve the modified Lagrangian problem as

$$\bar{f}_\mu(\lambda) = \lambda b + \max\{ (c - \lambda A)u - \mu d(u) : u \in U \} , \quad (6)$$

with an appropriately defined strongly convex *prox-function* $d(u)$ so that \bar{f}_μ is a smooth lower approximation of f , and the two minima can be related by a simple function of μ . Thus, one can apply a fast gradient to \bar{f}_μ and, appropriately managing μ , efficiently obtain an approximate solution to (1). This approach has been quite successful in several applications that require the solution of large-scale convex optimization problems [49], such as machine learning, data mining, inverse problems, and imaging (e.g., [1,17]). In turn, this has stimulated a vibrant research stream that is producing new results [11,44,7]. While the modification (6) is typically not viable in IP applications, *primal-dual SM* (PDSM) [54] can be defined that try to achieve similar results with an oracle for the original function. Indeed, the recent universal fast gradient method [55] automatically switches from the fast gradient, when f has the required properties, to PDSM when these are missing; for this reason, in this paper we take PDSM as the representatives of “modern” SM. Even the very recent [38], which combines in a unified framework PDSM with the Mirror-Descent method [51], provides only a slight generalization that does not significantly enlarge the class of approaches that can be implemented.

The aim of this paper is to assess how the recent developments in SM have influenced their computational significance for the approximate solution of LD of IPs. Our interest is motivated by the fact that, when used to provide lower bounds on (3), (1) has to be solved with the same accuracy required to the solution of (3), which is usually around $1\text{e-}4$ relative. This value is, broadly speaking, not so coarse that a SM is clearly the best choice to attain it (as would, say, be $1\text{e-}2$), but as well not so fine as to be basically hopeless to attain with a SM (as would, say, be $1\text{e-}6$). This middle ground needs therefore to be explored computationally. Towards that aim we unify most of the

known SM under a general scheme, starting from [21] that first unified deflection and projection and adding a number of other practically relevant issues such as several different forms of deflection and stepsize formulæ, incremental approaches, and dynamic generation of Lagrangian variables. The aim is not providing theoretical contributions: some of the variants that we have tested do not even have a rigorous convergence proof (cf. Table 4). We have instead developed an object-oriented C++ code, which we plan to openly release, that implements the proposed general scheme in a flexible way so as to make it easy to add new variants. The code is tested on the solution of two different LD of the Fixed-Charge Multicommodity Capacitated Network Design (FC-MCND) problem [18]. While both relaxations exhibit the block-separable form (4), they differ—for the same FC-MCND instance—in $|\mathcal{K}|$, n , and whether or not $A = \mathbb{R}^n$. These characteristics have an impact on the optimal choice of the algorithmic parameters for SM, helping in better characterizing the strengths and weaknesses of each variant. However, the two LD ultimately compute the same bound, which allows for an interesting comparison between them as well as with other solution methods that attain the same bound, such as different algorithms to solve the same LD and the use of general-purpose LP solvers.

The paper is organized as follows. In Section 2 we discuss the main characteristics of the SM presented in the literature, and we discuss a unified algorithmic scheme that encompasses them. Section 3 is dedicated to our extensive numerical experiments: we describe the target FC-MCND problem and its two different Lagrangian relaxations, then the experimental setup, and finally the results of the best SM variants, briefly comparing them with other approaches. These results, and the learned lessons, are summarized in Section 4. The Appendix contains the details of the algorithmic parameters of the SM we have used and of the tuning we have performed on them.

2 A general subgradient scheme

In this section we discuss the basic building blocks of SM, and we describe a general scheme encompassing many of the variants proposed in the literature.

2.1 Building blocks of subgradient methods

Each SM is constructed by combining a set of basic “building blocks”. We now briefly discuss them, with the fine details provided in the Appendix.

2.1.1 Stepsize rules

A crucial aspect of any SM is the selection of the *stepsize* ν_i . One of the surprising properties of these algorithms is that the stepsize can be in fact chosen without any knowledge, either a-priori or a-posteriori, of the specific

184 function to be minimized; indeed, any choice of the stepsize satisfying the
 185 so-called *diminishing/square summable* condition

$$\sum_{i=1}^{\infty} \nu_i = \infty \quad , \quad \sum_{i=1}^{\infty} \nu_i^2 < \infty \quad ,$$

186 of which $\nu_i = 1/i$ is the prototypical example, leads to a convergent algorithm.
 187 While this emphasizes the robustness of SM, these Stepsize Rules (SR) are
 188 most often inefficient in practice. The first efficient SR is due to Polyak [58],
 189 and simply reads $\nu_i = \beta_i(f_i - f_*)/\|g_i\|^2$, with $\beta_i \in (0, 2)$ arbitrary. This,
 190 however, needs to be revised, because in general $d_i \neq g_i$ (cf. §2.1.2), and f_* is
 191 not known. This leads to the Polyak-type *target value SR* of the form

$$\nu_i = \beta_i (f_i - f_i^{lev}) / \|d_i\|^2 \quad (7)$$

192 where f_i^{lev} is some approximation of f_* . Several SR of this type have been
 193 proposed; see, e.g., [5, 6, 12, 20, 43, 60]. Except in specific cases that will be
 194 discussed separately, all of our SR will have this form. The nontrivial issue in
 195 (7) is, clearly, how f_i^{lev} is determined. Without any external information, the
 196 typical approach is the *target following* one, where $f_i^{lev} = f_i^{rec} - \delta_i$ using the
 197 *record value* $f_i^{rec} = \min\{f_l : l = 1, \dots, i\}$ and the *displacement* $\delta_i > 0$ (which
 198 guarantees $\nu_i \geq 0$). The rules for choosing δ_i are divided into *vanishing* and
 199 *nonvanishing* ones according to the fact that $\delta_i \searrow 0$ as $i \rightarrow \infty$, or not [21, 42,
 200 59]. However, our application has the specific benefit that often a *lower bound*
 201 on f_* is available. This is typically provided by the cost $c\bar{u}$ of some feasible
 202 solution of (3). In theory \bar{u} may not always be available, for instance because
 203 (3) is actually empty. However, in many cases feasible lower bounds can easily
 204 be computed early on. For the application of §3.1, for instance, it is easy to
 205 detect if a solution exists at all by simply solving a continuous relaxation; if not
 206 there is no point in solving (1), otherwise rounding provides a feasible solution
 207 \bar{u} which can be used as a feasible lower bound to all nodes of a B&C approach.
 208 Indeed, at each node the algorithm is stopped as soon as $f_i^{rec} \leq c\bar{u}(1 + \eta)$,
 209 where η is the required *relative accuracy* for the solution of (3). Hence, in our
 210 tests we will assume that a lower bound $\underline{f} \leq f_*$ is available, which provides a
 211 workable f_i^{lev} without a need for target following techniques to be used. This
 212 allowed us to reduce the set of SR to be tested to only the following three:

- 213 1. the **Polyak** rule [58], whereby β_i and f_i^{lev} do not depend on i ;
- 214 2. the **ColorTV** rule as implemented in the **Volume** algorithm [5], which is
 215 based on classifying the iterations as *green*, *yellow* and *red* according to
 216 the improvement $\Delta f_i = f_{i-1} - f_i$ and the scalar product $d_{i-1}g_i$ (cf. §A.2);
- 217 3. the **FumeroTV** rule introduced in [31], specifically designed for LDs of IPs
 218 and that changes both β_i and f_i^{lev} in two different phases (cf. §A.2).

219 It would, however, be straightforward to test other approaches in our C++
 220 framework, such as the standard target following ones [21, 42, 59]. In fact, other
 221 than the above three Polyak-type rules, we have also tested the entirely dif-
 222 ferent SR corresponding to PDSM, as discussed next.

2.1.2 Deflection

We have always used the fairly (although not entirely) general version

$$d_i = \alpha_i g_i + (1 - \alpha_i) d_{i-1} \quad (8)$$

of the Deflection Rule (DR) (5) to compute the next iterate, for the *deflection parameter* $\alpha_i \in [0, 1]$. The use of a convex combination is crucial in the analysis, because it ensures that d_i is always an approximate (conditional, cf. §2.1.3) subgradient of f , as recalled in §2.2. Furthermore, this allows to produce (hopefully, asymptotically feasible) primal solutions $u \in \text{conv}(U)$ that are useful, e.g., for the active-set strategy discussed in §2.1.5. Finally, since d_i is ultimately to be scaled by the stepsize ν_i , the multipliers can always be scaled (up or down) as to sum to one, with the scaling factor then accounted by ν_i . For our experiments we have considered the following three DR:

1. the **STSubgrad** rule of the non-deflected SM [58], i.e., $\alpha_i = 1$;
2. the **Volume** rule where α_i is chosen as the (safeguarded) optimal solution of the one-dimensional quadratic problem [5,6] (cf. §A.3);
3. the **Primal-Dual** rule of [54] for PDSM, which actually chooses α_i and ν_i *simultaneously* in order to obtain optimal worst-case estimates on the SM convergence rate, in both the *Simple Averages* (SA) and the *Weighted Averages* (WA) variants (cf. §A.3).

Other rules have been proposed, such as the original one in [14] which used the largest possible α_i yielding $d_{i-1} g_i \geq 0$ (i.e., $\alpha_i = 1$ if the property already holds). Again, testing then in our C++ framework would be straightforward.

2.1.3 Projection

In the constrained case, what is actually minimized is the *essential objective* $f_A(\lambda) = f(\lambda) + \mathbf{1}(\lambda)$, where $\mathbf{1}(\cdot)$ is the (convex) indicator function of A (i.e., $\mathbf{1}(\lambda) = 0$ if $\lambda \in A$, and $\mathbf{1}(\lambda) = \infty$ otherwise). It is well-known that the *normal cone* N_i to A at λ_i , which is the polar of T_i , is $\partial \mathbf{1}(\lambda_i)$. Projecting g_i on T_i is then choosing some $w_i \in \partial \mathbf{1}(\lambda_i)$ in order to use $g_i + w_i \in \partial f_A(\lambda_i)$, instead of just g_i , to define d_i . While this is quite natural, at least if A is easy to project upon, things are more complex under (8), as there are then 8 possible deflection schemes, corresponding to all possible combinations to projecting g_{i-1} , d_{i-1} and d_i . The analysis of [21] shows that theoretical convergence can be attained in two ways. The first is the *stepsize-restricted* one, limited to stepsize rules of the form (7), which requires the satisfaction of the *safe rule*

$$\beta_i \leq \alpha_i (\leq 1) \quad , \quad (9)$$

ensuring that a step over a direction that is very far from $-g_i$ cannot be too large. In the *deflection-restricted* one, ν_i can rather be chosen arbitrarily provided that α_i is kept “large enough” by

$$(\nu_i \|d_{i-1}\|^2)(f_i - f_i^{lev} + \nu_i \|d_{i-1}\|^2) \leq \alpha_i \quad . \quad (10)$$

259 If projection on A is too expensive, it could be substituted with partial projec-
 260 tions working onto the individual constraints sets [16]. This would not change
 261 much the algorithmic scheme presented in this paper; besides, “complex” A
 262 are comparatively rare in our preferred application.

263 2.1.4 Incremental approaches

264 When $|\mathcal{K}|$ is very large, the total cost for computing $f(\lambda_i)$ may be large even
 265 if each f^k is, taken individually, quite inexpensive. Motivated by training ap-
 266 proaches for machine learning, *incremental* SM have been developed where
 267 the “full” subgradient g_i is replaced by g_i^k of one component $k \in \mathcal{K}$. Ideally, a
 268 sequence of *incremental* (inner) iterations performed along single-component
 269 subgradients could be roughly as effective as a sequence of *full* (outer) iter-
 270 ations, while the function evaluation cost is reduced by a factor of $1/|\mathcal{K}|$ [9,
 271 56]. However, to guarantee convergence one needs to regularly compute the
 272 whole function f , so not all the iterates can be incremental. Besides, due to
 273 the risk that a step along one “rogue” component may move λ_i away from
 274 the optimum, the stepsize of incremental iterations need to be reduced with
 275 respect to that of full ones (cf. (14) below).

276 2.1.5 Active set

277 When the number n of variables (i.e., of $Au = b$ constraints in (3)) is large,
 278 it may be convenient to employ an *Active Set* (AS) strategy whereby only a
 279 (small) subset of them is given a nonzero value at each iteration [26, 29, 30].
 280 This is in particular sensible if the constraints have the form $Au \leq b$ (\implies
 281 $A = \mathbb{R}_+^n$), because one can expect that only a fraction of them will actually be
 282 binding at optimality. Indeed, the AS allows to deal even with exponentially
 283 many constraints, provided that an efficient separator is available, which is
 284 known as “Relax-and-Cut” [35]. The relevant technical issue is what solution
 285 $u \in U$ is used to perform separation, i.e., to identify violated constraints to be
 286 added to the AS. An obvious choice is the optimal solution u_i of (4) for the
 287 current iterate λ_i , but a more sound choice is the *convexified solution* \bar{u}_i that
 288 can be generated at each iteration [2, 5, 6, 34, 46] and that, under appropriate
 289 conditions, converges to the optimal solution of (3) (if it is a convex problem,
 290 of its convexified relaxation otherwise). Under (8), this is simply obtained as
 291 $\bar{u}_i = \alpha_i u_i + (1 - \alpha_i) \bar{u}_{i-1}$. The AS technique poses little convergence issues if
 292 the AS is monotonically increasing (eventually, all variables will be active);
 293 careful removal of variables from the AS is also possible.

294 2.1.6 Summary

295 All these aspects give rise to a rather large set of possible combinations, many
 296 of which have algorithmic parameters that have to be tuned for optimal per-
 297 formances. Not all of these combinations have reliable proofs of convergence,
 298 although several do (cf. Table 4). In practice, barring dramatic mis-settings of

299 the algorithmic parameters, all the ones we have tested showed at least some
 300 degree of convergence, confirming the well-known fact that SM are remarkably
 301 robust approaches. Despite being very many (cf. the Appendix), the combi-
 302 nations that we have tested do not cover all possible variants of SM. Among
 303 the techniques that have been left out of the experiments are space-dilation
 304 methods [41, §7.2], other SR like variants of the Polyak stepsize [41, (7.11)]
 305 or Ermoliev-like stepsizes [41, (7.6)–(7.9)], the *heavy ball* SM [62] popular in
 306 machine learning, and others. Yet, the structure of our C++ code would allow
 307 to easily incorporate most of these variants.

308 2.2 A generic subgradient scheme

309 We now present a generic scheme of SM, in order to be able to discuss the
 310 nontrivial interactions between its individual components.

```

0. Input the algorithmic parameters, among which StepRes;
   Select  $\bar{\lambda}_0 \in \Lambda$ ;  $\lambda_1 \leftarrow \bar{\lambda}_0$ ,  $\bar{f}_0 = -\infty$ ,  $d_0 \leftarrow 0$ ,  $i \leftarrow 0$  and go to step 4;
1. Possibly,  $d_{i-1} \leftarrow P_{T_i}(d_{i-1})$ ;
   if( StepRes ) then  $\alpha_i = \text{Deflection}()$ ;  $\text{ComputedD}()$ ;  $\nu_i = \text{Stepsize}(\alpha_i)$ ;
   else  $\nu_i = \text{Stepsize}()$ ;  $\alpha_i = \text{Deflection}(\nu_i)$ ;  $\text{ComputedD}()$ ;
2. If some stopping test is satisfied, exit;
3.  $\lambda_{i+1} \leftarrow P_\Lambda(\bar{\lambda}_i - \nu_i d_i)$ ;
4. OutItr = true if an outer (full) iteration is to be performed;
   if( OutItr ) then evaluate  $f_{i+1} = f(\lambda_{i+1})$  and  $g_{i+1} \in \partial f(\lambda_{i+1})$ ;
   else select  $k$ , evaluate  $f^k(\lambda_{i+1})$  and  $g_{i+1} \in \partial f^k(\lambda_{i+1})$ ;
5. Possibly,  $g_{i+1} \leftarrow P_{T_i}(g_{i+1})$ . Select  $\bar{\lambda}_{i+1}$ , set  $\bar{f}_{i+1}$  accordingly;
6.  $i \leftarrow i + 1$  and go to step 1.

```

312 The following general remarks discuss the common features of all the variants.

- 313 – The new iterate is generated at Step 3 starting from the *stability center*
 314 $\bar{\lambda}_i$, which is updated at Step 5. In the original SM the update is always
 315 $\bar{\lambda}_{i+1} = \lambda_{i+1}$. In the parlance of Bundle methods, this is called a *Serious*
 316 *Step* (SS), as opposed to *Null Steps* (NS) where $\bar{\lambda}_{i+1} = \bar{\lambda}_i$. Changing $\bar{\lambda}_i$ is
 317 sensible if this leads to a significant improvement of the function value, i.e.,
 318 $\Delta f_i = \bar{f}_i - f_{i+1} \gg 0$, otherwise a NS may be preferable. This is the strategy
 319 often used (cf. §A.3), although PDSM provide an entirely different rationale
 320 for using a stability center, without ever changing it. In our implementation
 321 either a SS or NS is always performed, as all SM variants we are aware of
 322 only employ these (whereas Bundle methods exist that can make different
 323 choices [3]). All this requires some quite obvious changes in some of the
 324 standard formulæ, such as using \bar{f}_i instead of f_i in (7) and (10).
- 325 – The `ComputedD()` subroutine extends (8) to $d_i = \alpha_i \bar{g}_i + (1 - \alpha_i) \bar{d}_{i-1}$, where
 326 \bar{g}_i and \bar{d}_{i-1} are either g_i and d_{i-1} or their projection over the *tangent cone*
 327 T_i of Λ at the *stability center* $\bar{\lambda}_i$. Furthermore, possibly $d_i \leftarrow P_{T_i}(d_i)$,

yielding all 8 possible deflection schemes. Yet, since T_i is convex, if both g_i and d_{i-1} are projected then $d_i \in T_i$ already, thus projecting is avoided.

- The (fixed) algorithmic parameter `StepRes` controls whether ν_i is computed after d_i (stepsize-restricted) or vice-versa (deflection-restricted). Since computing d_i requires α_i , `Computed()` always comes after `Deflection()`. However, in the deflection-restricted approach, the safe rule (9) requires ν_i in order to choose α_i , and consequently `Stepsize()` has also to be called before `Computed()`. Note that, in this case, (7) would require $\|d_i\|$ before having computed d_i , which is then replaced by $\|d_{i-1}\|$. The stepsize-restricted case is more natural for (7) in that d_i is computed before ν_i is. In PDSM, ν_i and α_i are chosen simultaneously, and therefore `StepRes` has no effect. Since we do not restrict ourselves to theoretically convergent methods, we also allow to switch off the safe rules (9)/(10).
- To update the AS (if any), the primal solution \bar{u}_i (cf. §2.1.5) is needed, which depends on the choice of α_i . Hence, the AS can only be updated after that `Deflection()` has been called. However, if the AS changes, then the vectors d_{i-1} and g_i need to be updated to take into account the new components, which in turn may change α_i . Hence, after an AS update, we compute again the deflection parameter α_i , and in the deflection-restricted scheme also the stepsize; the process is repeated until the AS remains unchanged. Also, projection on T_i should be re-done each time new variables are added. However, with $A = \mathbb{R}_+^n$ the projection can be computed component-wise, hence only the new components of d_{i-1} , g_i and/or d_i need be dealt with.
- The *linearization error* of g_i at $\bar{\lambda}_i$ is

$$\sigma_i = \sigma_i(\bar{\lambda}_i) = \bar{f}_i - [f_i + (\bar{\lambda}_i - \lambda_i)g_i] = \sigma_i(\bar{\lambda}_{i-1}) - \Delta\bar{f}_i - (\bar{\lambda}_i - \bar{\lambda}_{i-1})g_i \quad , \quad (11)$$

where $\Delta\bar{f}_i = \bar{f}_{i-1} - \bar{f}_i$. Note that $\Delta\bar{f}_i \neq \Delta f_i$ when a NS occurred at iteration $i-1$, i.e., $\bar{\lambda}_i = \bar{\lambda}_{i-1} \implies \Delta\bar{f}_i = 0$. Convexity of f ensures that $\sigma_i \geq 0$ and $g_i \in \partial_{\sigma_i} f(\bar{\lambda}_i)$. Furthermore, σ_i can be easily kept updated when $\bar{\lambda}_i$ changes using (11), which is useful since it may play a role at different points in the algorithm, such as some of the DR (cf. §A.3) and the stopping tests (cf. next point). However, when projection is used, one rather wants to compute the linearization error of the *projected* $\bar{g}_i \in \partial[f + \mathbb{1}](\bar{\lambda}_i)$. This is why the projection of g_i is not performed at Step 1, but it occurs before updating $\bar{\lambda}_i$ at Step 5: so that, in case of a SS, the linearization error of \bar{g}_i is computed. A downside of this choice is that if $\bar{\lambda}_i$ changes at Step 5, then g_i may have to be projected again in the next iteration; however, projections (if at all required) are inexpensive in our applications.

- An advantage of (8), which underlines all the analysis in [21], is that we can similarly compute and keep updated the linearization error of d_i w.r.t. $\bar{\lambda}_i$. That is, knowing that $d_{i-1} \in \partial_{\epsilon_{i-1}} f(\bar{\lambda}_i)$, one has $d_i \in \partial_{\epsilon_i} f(\bar{\lambda}_i)$ with $\epsilon_i = \epsilon_i(\bar{\lambda}_i) = \alpha_i \sigma_i(\bar{\lambda}_i) + (1 - \alpha_i) \epsilon_{i-1}(\bar{\lambda}_i)$. Also, ϵ_i can be cheaply updated after a SS with $\epsilon_i(\bar{\lambda}_{i+1}) = \epsilon_i(\bar{\lambda}_i) - \Delta\bar{f}_{i+1} - (\bar{\lambda}_{i+1} - \bar{\lambda}_i)d_i$. This means, however, that the same issue about projection arises here also.

371 – In the un-deflected SM, it is possible to use the inverse of $\|g_i\|$ in (7)
 372 because as soon as $\|g_i\| = 0$, one has proven the optimality of λ_i . Since
 373 $g_i \in \partial_{\sigma_i} f(\bar{\lambda}_i)$, this also means that $\bar{\lambda}_i$ is σ_i -optimal. With the provisions
 374 above, the same holds for d_i (or its projection); that is one can stop when
 375 both $\|d_i\|$ and ϵ_i are “small”. Our particular implementation is

$$t^* \|d_i\| + \epsilon_i \leq \eta \max(1, |f_i^{rec}|) \quad (12)$$

376 where t^* is an appropriately chosen “large” scaling factor [25] and η is the
 377 required final relative accuracy (typically, $\eta = 1\text{e-}4$).

378 – As suggested in [54] (and in [3] in a different context), one could also use
 379 the deflection parameter α_i in a different way: not to change the gradient,
 380 but the point where it is evaluated. That is, for the recursive formulæ

$$\hat{\lambda}_i = \alpha_i \lambda_i + (1 - \alpha_i) \hat{\lambda}_{i-1} \quad , \quad \hat{f}_i = \alpha_i f_i + (1 - \alpha_i) \hat{f}_{i-1}$$

381 with $(\hat{\lambda}_0, \hat{f}_0) = (0, 0)$, one has $\hat{f}_i \geq f(\hat{\lambda}_i)$ for all i , and therefore an approx-
 382 imation of the linearization error of d_i with respect to $\hat{\lambda}_i$ is

$$\hat{\epsilon}_i = \hat{\epsilon}_i(\hat{\lambda}_i) = \alpha_i \hat{\sigma}_i(\hat{\lambda}_i) + (1 - \alpha_i) \hat{\epsilon}_{i-1}(\hat{\lambda}_i)$$

383 (with $\hat{\epsilon}_1(\hat{\lambda}_1) = \hat{\sigma}_1(\hat{\lambda}_1)$ and $\hat{\sigma}_i(\hat{\lambda}_i) = \hat{f}_i - [f_i + (\hat{\lambda}_i - \lambda_i)g_i] = (1 - \alpha_i)[\hat{f}_{i-1} -$
 384 $f_i - (\hat{\lambda}_{i-1} - \lambda_i)g_i]$). Hence $d_i \in \partial_{\hat{\epsilon}_i} f(\hat{\lambda}_i)$ for all i , which allows to also
 385 employ the alternative stopping criterion

$$t^* \|d_i\| + \hat{\epsilon}_i \leq \eta \max(1, |f_i^{rec}|) \quad . \quad (13)$$

386 Testing (13) is free in PDSM, since all the terms involved have to be com-
 387 puted anyway (cf. §A.3). For all the other approaches we only used (12),
 388 for again in most cases $\|d_i\|$ and ϵ_i are required anyway in the SR and/or
 389 the DR. However, both stopping conditions are hardly if ever satisfied in
 390 practice, and typically the algorithm stops at the pre-set iterations limit.

391 – At Step 4, some logic is used to decide whether an outer (full) iteration
 392 is computed, thereby evaluating all the components, or only one compo-
 393 nent is evaluated. This is done in a simple pattern: we perform one outer
 394 iteration, followed by $|\mathcal{K}| + 1$ inner iterations, one for each of the different
 395 components plus one for the linear component corresponding to the RHS.
 396 As suggested in [9, 56], we randomize the order in which the components
 397 are chosen, with the random permutation changed at every outer iteration.
 398 We experimented with different ratios between inner and outer iterations
 399 but the results were inconclusive, with the simple approach being in general
 400 the best one. Furthermore, this means that a group of $|\mathcal{K}| + 2$ consecutive
 401 iterations (one outer, the other inner) costs, at least as far as the subprob-
 402 lem solution is concerned, as much as two full iterations. This is useful
 403 when comparing the running time of the approaches, as discussed in §3.2.
 404 When the AS strategy is used we update the AS only at full iterations,
 405 since its cost is comparable to that of one full iteration (cf. again §3.2),
 406 and doing it more frequently would largely negate the advantage of having
 407 faster iterations. Updating the active set less frequently is possible, but it
 408 has not shown to be computationally convenient in our application.

409 – In the incremental SG, deflection is never used ($\alpha_i = 1$); there is no theo-
 410 retical support for deflecting the inner steps, and also how to deflect outer
 411 ones is unclear. For inner steps, (7) would require to compute the norm
 412 of $g_i \in \partial f(\lambda_i)$, but only g_i^k for one $k \in \mathcal{K}$ is available. Following [50] we
 413 replace $\|g_i\|$ by the global Lipschitz constant L of f , yielding

$$\nu_i = \beta_i \frac{\bar{f}_{p(i)} - f_i^{lev}}{\chi |\mathcal{K}| L^2} \quad (14)$$

414 where $p(i)$ the last outer step before i and χ is an arbitrary constant. In
 415 other words, one keeps the main part of the stepsize unchanged during
 416 sequences of inner iterations between two outer ones. In the same vein in
 417 our experiments we used $\beta_i = \beta_{p(i)}$ and $f_i^{lev} = f_{p(i)}^{lev}$.

418 3 Numerical experiments

419 We now present our extensive computational experiments on two different La-
 420 grangian relaxations of the Fixed-Charge Multicommodity Capacitated Net-
 421 work Design (FC-MCND) problem [18], rapidly recalled below.

422 3.1 Lagrangian relaxations for FC-MCND

Given a directed network $G = (N, A)$, we must satisfy the demands of a set
 of *commodities* K . Each $k \in K$ is an origin-destination pair (s_k, t_k) with an
 associated demand $d^k > 0$ that must flow between them, i.e., a deficit vector
 $b^k = [b_i^k]_{i \in N}$ such that $b_i^k = -d^k$ if $i = s_k$, $b_i^k = d^k$ if $i = t_k$, and $b_i^k = 0$
 otherwise. Each arc $(i, j) \in A$ can only be used, up to its mutual capacity
 $u_{ij} > 0$, if the corresponding fixed cost $f_{ij} > 0$ is paid. Also, individual
 capacities u_{ij}^k are imposed for each commodity k . Finally, the routing cost c_{ij}^k
 has to be paid for each unit of commodity k on (i, j) . FC-MCND consists
 in minimizing the sum of all costs while satisfying demand requirements and
 capacity constraints, its classical arc-flow formulation being

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \quad (15)$$

$$\sum_{(j,i) \in A} x_{ji}^k - \sum_{(i,j) \in A} x_{ij}^k = b_i^k \quad i \in N, k \in K \quad (16)$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij} \quad (i, j) \in A \quad (17)$$

$$x_{ij}^k \leq u_{ij}^k y_{ij} \quad (i, j) \in A, k \in K \quad (18)$$

$$x_{ij}^k \geq 0 \quad (i, j) \in A, k \in K \quad (19)$$

$$y_{ij} \in \{0, 1\} \quad (i, j) \in A \quad (20)$$

423 For our tests we have employed two Lagrangian relaxations of FC-MCND.
 424 In the first one relaxes constraints (17)–(18) with multipliers $\lambda = [\alpha, \beta] =$

425 $[\alpha_{ij}, \beta_{ij}^k]_{(i,j) \in A, k \in K} \geq 0$, yielding the objective function

$$\min \sum_{(i,j) \in A} \sum_{k \in K} (c_{ij}^k + \alpha_{ij} + \beta_{ij}^k) x_{ij}^k + \sum_{(i,j) \in A} (f_{ij} - \alpha_{ij} u_{ij} - \sum_{k \in K} u_{ij}^k \beta_{ij}^k) y_{ij}$$

426 whose minimization subject to the remaining (16), (19)–(20) reduces to $|K|$
 427 shortest path problems, plus $|A|$ trivial single-variable IPs. This justifies the
 428 name “Flow Relaxation” (FR), although what is relaxed are rather knapsack-
 429 type constraints. Since (16), (19) only involve continuous variables, the LD pro-
 430 vides the same bound as the continuous relaxation. Note that the constraints
 431 (18) are many; these being inequalities, this is the setting where AS techniques
 432 can be expected to be effective [29]. An estimate of the Lipschitz constant,
 433 useful for the incremental SM (cf. (14)) as well as for PDSM (cf. (25)), is
 434 $L = \sqrt{\sum_{(i,j) \in A} (u_{ij})^2 + \sum_{k \in K} \sum_{(i,j) \in A} (u_{ij}^k)^2}$. Note that when the AS is used
 435 the capacities entering the above formula are only those of the constraints in
 436 the AS, and therefore L changes as the algorithm proceeds.

437 In the second relaxation one rather dualizes the flow conservation con-
 438 straints (16) with multipliers $\lambda = [\lambda_i^k]_{i \in N, k \in K}$, yielding the objective function

$$\min \sum_{(i,j) \in A} (\sum_{k \in K} (c_{ij}^k + \lambda_i^k - \lambda_j^k) x_{ij}^k + f_{ij} y_{ij}) \quad [+ \sum_{i \in N} \sum_{k \in K} \lambda_i^k b_i^k]$$

439 whose minimization subject to the remaining (17)–(20) basically decomposes
 440 into $|A|$ continuous knapsack problems, one to determine the optimal value of
 441 each integer variable y_{ij} . This justifies the name Knapsack Relaxation (KR),
 442 although what is relaxed are flow conservation constraints. It can be shown
 443 that, due to (18), the relaxation has the integrality property: hence, as in the
 444 previous case the LD gives the same bound as the continuous relaxation. The
 445 number of multipliers is still rather large; however, these being equalities, it
 446 is unlikely that many of them are not going to be active at optimality, and
 447 therefore the AS technique is less likely to be effective. Unlike in the FR,
 448 there are no sign constraints on the multipliers, and therefore no projection
 449 is needed. The Lipschitz constant is $L = \sqrt{\sum_{k \in K} \sum_{i \in N} (L_i^k)^2}$, where $L_i^k =$
 450 $\max[| -b_i^k + \sum_{(j,i) \in A} u_{ji}^k |, | -b_i^k - \sum_{(i,j) \in A} u_{ij}^k |]$.

451 Note that, being (15)–(20) a minimization problem (unlike (3)), both LD
 452 are maximization problems (unlike (1)). This is easily catered in the implemen-
 453 tation by changing the sign of the objective function and of the subgradients.

454 3.2 Experimental setup

455 We have implemented all the variants of SM within a general C++ framework for
 456 nonsmooth optimization developed by the authors along the years. The frame-
 457 work is based on two pure virtual classes, `NDOSolver` and `FiOracle`, which es-
 458 tablish the interface between the optimization algorithm (in our case, the SM
 459 implemented in the class `NDOSolver::Subgradient`) and the oracle computing
 460 f (in our case, the classes `FiOracle::FlowFiOrcl` and `FiOracle::KnapFiOrcl`
 461 for FR and KR, respectively). Other implementations of nonsmooth approaches,

such as different forms of Bundle methods [3,25,29], were already available within the framework. The `Subgradient` class in turn relies on two external classes, `Stepsize` and `Deflection`, so that the different SR (cf. §2.1.1) and DR (cf. §2.1.2) can be implemented as derived classes from these. The PDSM case, where ν_i and α_i are set together, is easily accounted for by having the corresponding `Primal-Dual` class to derive from *both* `Stepsize` and `Deflection`. This shows that while the general scheme depicts the two aspects as independent, there is no problem when they actually have to be synchronized. Moreover, the code is designed for dealing with more complex A requiring projection on knapsack-like constraints by means of the `CQKnPClass` class [28]. The code has been compiled with GNU `g++ 4.4.5` (with `-O3` optimization option) and ran single-threaded on an Opteron 6174 processor (12 cores, 2.2 GHz) with with 32 GB of RAM, under a i686 GNU/Linux operating system. To solve the FR, we have used solvers from the `MCFCClass` project, available at <http://www.di.unipi.it/optimize/Software/MCF.html>, while solving the KR basically just required a sort and was coded directly. When comparing SM with other approaches we used `Cplex 12.5.0.1` to solve LPs.

The numerical experiments have been performed on 80 randomly generated instances, arranged in 20 groups of 4 instances each. The first 8 groups are of small size. In the remaining 12 groups the number of nodes and arcs are chosen as (20, 300), (30, 600), or (50, 1200), and for each of these $|K|$ is chosen in {100, 200, 400, 800} (cf. Table 1). We refer to [29] for more details; the instances can be downloaded from <http://www.di.unipi.it/optimize/Data/MMCF.html>.

A nontrivial issue about our experiments is how to compare the performances of the different SM. Our choice has been to record the running time and the obtained lower bound of each variant with different iteration count limits. For all non-incremental SM, we (somewhat arbitrarily) choose that to be 100, 200, 500, 1000, 2000, 5000, and 10000 iterations. For incremental SM, whose inner iterations are faster, the iteration counts of 1000, 2000, 5000, 10000, 20000, 50000, 100000, 200000, 500000 and 1000000 were used instead. We then charted the time required to reach a certain gap with the (known) optimal value. An issue with this approach is that computing the f value in instances of larger size is more costly, making it difficult to compute aggregated results. Fortunately, for our instances a simple scaling was sufficient. Indeed, we observed that the charts for the same SM variant and different sizes were remarkably similar, and they became almost identical by expressing them in *normalized running time*, i.e., dividing the running time by $|A| \cdot |K|$. This is reasonable because in both relaxations the computation of f is $O(|A| \cdot |K|)$ up to logarithmic factors ($|K|$ shortest paths with non-negative arc costs, hence $O(|A| \log(|N|))$ each, vs. $|A|$ continuous knapsack problems, hence $O(|K| \log(|K|))$ each), and, given the limited range of $|A|$ and $|K|$, any logarithmic factor is almost constant. All the rest of the algorithm has a linear cost in the number of variables n , which is $(|A|+1) \cdot |K|$ for the FR and $|N| \cdot |K|$ for the KR, but $|A|$ is proportional to $|N|$ as the graphs are sparse. With the AS strategy n is actually (much) smaller, but identification of new violated constraints is again $O(|A| \cdot |K|)$. All in all, the iteration cost is dominated by

508 factors of roughly $O(|A| \cdot |K|)$, explaining why the running time scales pretty
 509 much linearly in that quantity. It is also remarkable that the convergence speed
 510 proved to be very similar as n varied by orders of magnitude (from 9040 to
 511 960000 for the FR and from 800 to 40000 for the KR). This is not surpris-
 512 ing, since the theoretical efficiency estimates of SM are typically independent
 513 on n ; our experiments confirm that the practical behaviour is in fact pretty
 514 much invariant with n , hence that SM can be especially promising for very
 515 large-scale problems. This allowed us to compare the different SM variants
 516 by comparing their convergence graphs aggregated across *all* the 80 instances
 517 of our test set. Note that incremental variants actually are randomized al-
 518 gorithms due to the selection of the random reshuffle of the components at
 519 each full iteration; however, since each graph aggregates results among many
 520 instances, it is not necessary to repeat individual runs several times. All this
 521 has been instrumental in allowing us to perform the extensive tuning phase,
 522 detailed in §A.3, which led to the identification of the best results described
 523 in the next paragraph.

524 A final relevant aspect of our computational tests concerns the fact that the
 525 stepsize rules (7)/(10) require some (lower) approximation \underline{f} to f_* . In order to
 526 avoid target-level approaches we have worked with a fixed \underline{f} . However, in order
 527 to cater for the different cases that would occur when using these techniques
 528 in IP, we have used two different configurations: in one $\underline{f} = f_*$, and in the
 529 other $\underline{f} = f_* - 0.1|f_*|$. We denote the latter by “10% f_* ”; it corresponds to the
 530 case where the best known solution to (3) is 10% more costly than the best
 531 possible lower bound (somewhat on the “bad” side, but unfortunately not too
 532 unlikely), so that even if f_* were reached, the corresponding node in the B&C
 533 tree could not be fathomed. The case $\underline{f} = f_*$ is instead the one where the node
 534 can be fathomed by the bound, if the latter is computed accurately enough.

535 3.3 Results for the FR

536 We now report the numerical results of SM on the FR, using the best param-
 537 eters detailed in the §A.3. Each variant is represented in Figures 1 and 2 by
 538 means of a graph, with *normalized* total time (cf. §3.2) on the horizontal axis
 539 and *average gap* on the vertical one, both in logarithmic scale. We separately
 540 report results for all combinations of the three variants of SR and the two
 541 variants of DR (STSubgrad “(s)” and Volume “(v)”). We also report all SR
 542 with the incremental approach “(i)” (with no deflection, cf. §2.2), and the two
 543 SA and WA variants of PDSM. For clarity, we divide both Figures in four
 544 different quadrants, with the same scale on both axes to allow for comparison.
 545 The upper two graphs (part (a)) depict results when the AS strategy is used,
 546 and the lower two ones (part (b)) when it is not. The leftmost graphs depict
 547 the approaches when deflection is used (Volume and Primal-Dual) and the
 548 rightmost ones these where it is not (STSubgrad and incremental). Figure 1
 549 reports the results with $\underline{f} = f_*$, while Figure 2 those with $\underline{f} = 10\%f_*$; since
 550 PDSM do not use \underline{f} , the corresponding curves are the same in the two Fig-

551 ures. We did not report the performances of incremental approaches without
 552 the AS strategy because it was exceedingly slow. This is not surprising, be-
 553 cause in the FR just forming the whole subgradient has a cost comparable to
 554 that of solving *all* the subproblems, thereby negating any advantage in having
 555 incremental iterations.

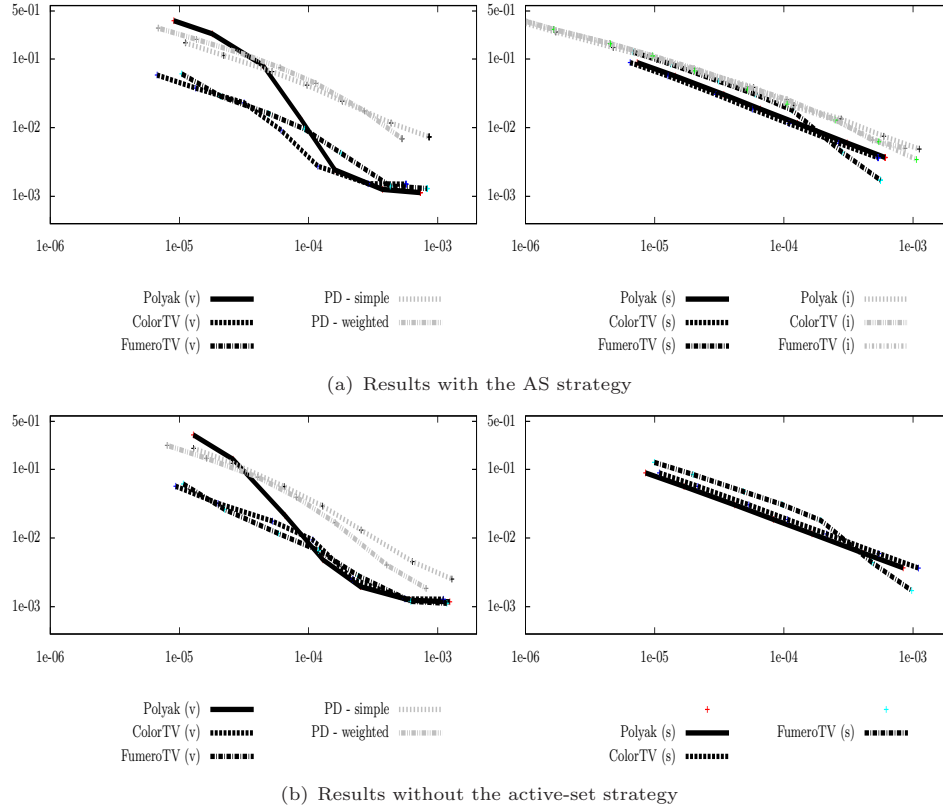


Fig. 1 Results for the FR with lower bound f_* (normalized time vs. average gap)

556 The following remarks can be made about the results.

- 557 – Deflected approaches are much more efficient than non-deflected ones, as
 558 can be seen by comparing the same SR (left vs. right graphs). This requires
 559 properly choosing how to deflect and which vectors among d_i , d_{i-1} and g_i
 560 is better to project. However, as discussed in §A.4, the different forms
 561 of projection have a limited impact on the performances, as long as any
 562 projection is performed, so deflection is most definitely the way to go.
- 563 – Incremental approaches are not competitive, which is likely due to the combi-
 564 nation of two factors. On the one hand, they are not deflected (cf. above).
 565 On the other hand n is large, so that just forming g_i^k requires much more
 566 time than computing f^k . Thus, each iteration has a large “fixed cost”,
 567 independent on how many components are computed, besides that of comput-
 568 ing f . While the AS strategy manages to decrease this cost, it is still not

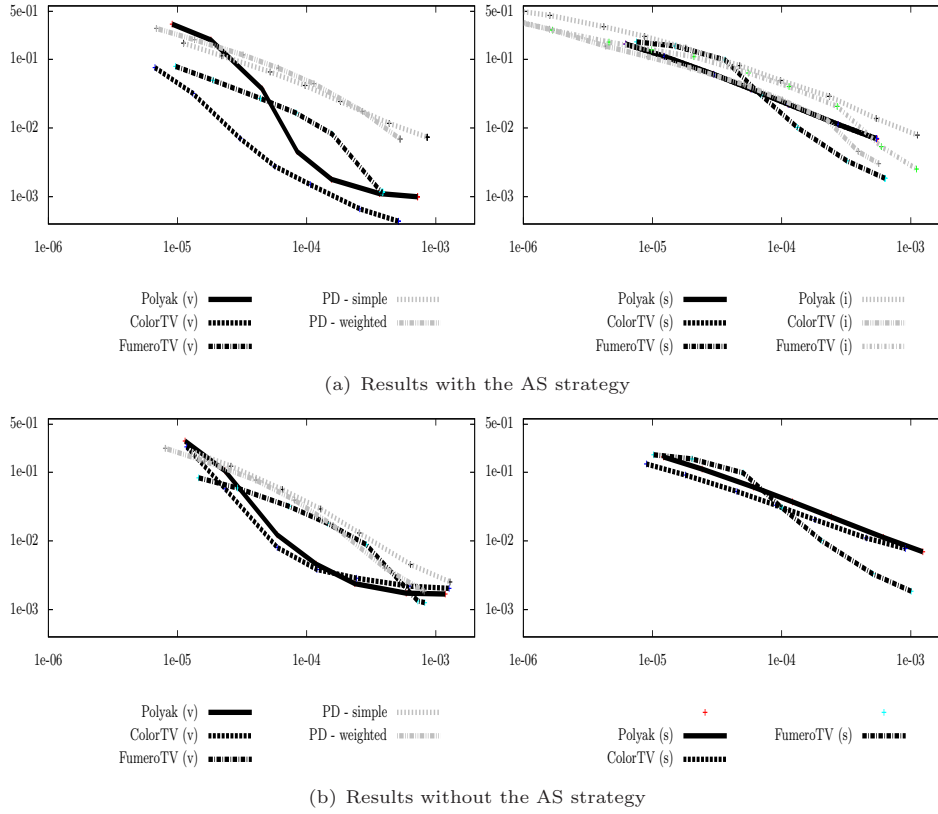


Fig. 2 Results for the FR with lower bound $10\%f_*$ (normalized time vs. average gap)

569 enough to make the incremental approach competitive. For this to happen
 570 n should be “small” w.r.t. the cost of computing each f^k , although if the
 571 latter is very large then other approaches may be preferable, cf. §3.5.

572 – PDSM are most often not competitive, although their convergence is very
 573 stable. The WA is typically better than the SA, as the theory suggests.
 574 PDSM can still be considered attractive in view of the very limited effort
 575 required to tune them; yet, finely tuned SM with other DR and SR can be
 576 significantly more effective. This may be partly due to the fact that PDSM
 577 do not use any available information about f^* , while (7)/(10) do. We also
 578 experimented with providing PDSM other information about the optimal
 579 solution to (1) (cf. §A.3), but with no success.

580 – The AS technique is in general beneficial: SM are somewhat faster in per-
 581 forming the same number of iterations (the topmost graphs in both Figures
 582 terminate more on the left than the bottom ones), while the convergence
 583 rate is usually similar. There are, however, exceptions. For instance, in
 584 “(v)” SM the AS can actually improve convergence speed (especially in
 585 Figure 2), while the converse happens for PDSM. This is not surprising
 586 since, to the best of our knowledge, AS techniques in the PSDM have never
 587 been analyzed; this may suggest that some specific theoretical development

may be useful in practice.

3.4 Results for the KR

The results of the KR are summarized in Figure 3, with largely the same notation as for the FR case. However, in this case the AS technique is not used, so only one figure is needed: part (a) is for $\underline{f} = f_*$, while part (b) is for $\underline{f} = 10\%f_*$. Since PDSM do not use \underline{f} , the corresponding curves are identical.

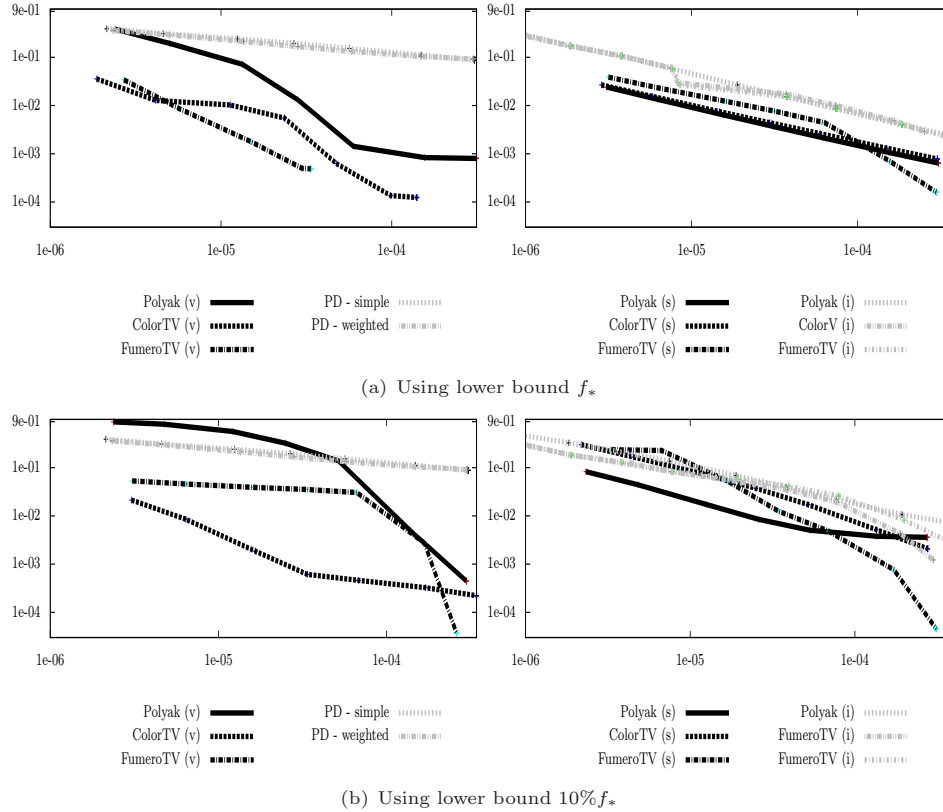


Fig. 3 Results for the KR (normalized time vs. average gap)

The following remarks can be made about the results:

- By and large, the same trends seen in the FR case show up here in terms of strong benefits of deflection and no benefits of incremental approaches.
- PDSM are even less competitive. This may be due to the fact that they have been developed under some sort of compactness assumption on \mathcal{A} (cf. (21)), and actually use its (estimated) diameter in setting the algorithmic parameters. In the KR, not only the feasible set is unbounded (this was true for the FR as well); since the relaxed constraints (16) are rank-deficient, the *set of optimal solutions* is also unbounded. This seems to

significantly affect the practical behaviour of PDSM.

- Figure 3(a) for $\underline{f} = f_*$ shows a peculiar behaviour of the `FumeroTV` rule: while it is the most efficient as it runs, it stops far before the maximal iteration limit because ν_i become too small, thereby getting a far worse final gap than the other variants (although quickly). This seems to be an issue with the rule, and no choice of the parameters we tested was able to avoid it. Interestingly, this only happens with deflection: it does not with `STSubgrad`, nor with $\underline{f} = 10\%f_*$. It may be possible that some settings that we have not tested may avoid this behaviour, but we elected to keep this as a cautionary tale about the fact that heuristic rules, while possibly working well in many cases, may fail sometimes.
- The convergence graph of `ColorTV` is noticeably shorter than the others (save for `FumeroTV`), as it often attains the required gap of $1\text{e-}4$ against the *known* lower bound f_* , at which point it is stopped. This can actually happen in the IP application, since $f_* < c\bar{u}$ may happen (the B&C node can be fathomed by the bound), which is particularly useful because the standard stopping rules (12)/(13) are scarcely effective.
- In general, the KR provides better bounds more quickly than the FR, confirming previous experiences [19].

3.5 Comparison with Cplex and Bundle methods

We now compare the best SM with two other approaches which provide the very same bound: solving the LP relaxation of (15)–(20) with a general-purpose LP solver, and solving the LD of the FR and the KR with a Bundle method. The experiments were performed as follows:

- For `Cplex`, an optimality gap of $1\text{e-}6$ has been set, and always attained. Tuning also has been performed by testing all of the available LP algorithms and selecting the dual simplex one, which provided the best performances; it was, however, almost always the algorithm chosen by the “automatic” setting. Also, the (many) constraints (18) have been introduced in the formulation as *lazy constraints*—the equivalent of using the AS strategy in SM—which was crucial for performances (cf. [29, Table 4]). We experimented with passing f_* to `Cplex`; since a dual simplex method is used, this might have allowed `Cplex` to stop as soon as a(n approximately) dual solution is achieved. However, this turned out to be of no use, precisely due to lazy constraints: `Cplex` separates them only when a feasible primal solution is attained, which is only at the end of the dual simplex. Not using the lazy constraints allowed `Cplex` to stop sooner when the information was provided, but it resulted in a hugely increased running time. By contrast, the other algorithms use infeasible primal solutions to do separation, and therefore do not suffer from this issue.
- For the Bundle method [25] a gap of $1\text{e-}4$ was required, although, unlike with SM, requiring a higher accuracy may only come at the cost of a comparatively minor increase in running times [29, Table 3 and Table 6].

The Bundle algorithm was also provided with f_* , which it uses both to stop as soon as a solution with accuracy $1e-4$ is attained and to improve the *cutting plane model* it uses to drive the search. We used two different variants of the Bundle method for the two LD. For the FR we used the fully disaggregated version with “easy component” and linear stabilization, denoted by DE-L in the table, that has been proven in [29]—after extensive tuning—to be the best option. It requires a costly master problem, which takes by far the largest fraction of running time, but it attains the desired solution in a very small number of iterations. For the KR, after extensive tuning (not discussed here in details) we found the best Bundle variant to rather be the one that uses a fully aggregated master problem with quadratic stabilization (denoted by AK-Q in the table), where the master problem is solved with the specialized QP solver of [24].

- For SM, we report results corresponding to the best options identified in the previous phase. In particular, for the FR we have used `Volume` as DR and `Polyak` as SR (denoted by FVP in the table), with the AS strategy, while for the KR we have used `Volume` as DR, but `ColorTV` as the SR (denoted by KVC in the table). For both algorithms, we have set $\underline{f} = f_*$, and required a gap of $1e-4$. We also set an iteration limit of 5000, as it seemed to represent the best compromise between accuracy of the achieved solution and running time. FVP invariably stopped at the iteration limit, so we only report the final gap. KVC instead often—but not always—reached $1e-4$ accuracy before the iteration limit, thus we report both the number of iterations and the final gap.

#	dimension			Cplex	FVP			KVC			DE-L		AK-Q	
	$ N $	$ A $	$ K $	time	time	gap	time	iter	gap	time	iter	time	iter	
1	20	226	40	0.05	1.76	1e-3	0.12	881	9e-5	0.09	12	0.25	1233	
2	20	230	200	17.71	11.07	2e-3	5.39	4738	1e-4	16.34	30	10.44	8084	
3	20	292	40	0.05	2.17	1e-3	0.10	602	1e-4	0.09	10	0.12	480	
4	20	292	200	16.42	14.12	1e-3	6.08	4604	1e-4	12.54	28	8.50	5225	
5	30	519	100	9.48	16.53	2e-3	3.15	3709	2e-4	10.05	34	8.05	7073	
6	30	519	400	191.30	87.07	1e-3	20.62	4631	1e-4	80.28	25	57.42	6713	
7	30	684	100	7.04	24.85	2e-3	3.27	3141	1e-4	10.90	53	5.03	3499	
8	30	692	400	450.36	125.89	1e-3	26.16	4903	2e-4	188.33	32	82.67	9830	
9	20	300	100	5.73	10.21	3e-3	2.52	5000	2e-4	7.36	35	3.62	5181	
10	20	300	200	26.62	24.29	1e-3	6.65	5000	2e-4	19.96	30	10.10	6083	
11	20	300	400	42.95	46.54	1e-3	17.45	4051	1e-4	16.77	26	38.18	5920	
12	20	300	800	148.35	107.66	1e-3	25.42	3538	1e-4	38.32	23	33.76	3232	
13	30	600	100	18.68	23.78	1e-3	6.13	4708	2e-4	7.93	42	11.16	6496	
14	30	600	200	50.89	44.94	9e-4	14.09	3368	1e-4	8.93	34	25.59	3896	
15	30	600	400	104.10	101.11	8e-4	20.98	3208	1e-4	11.51	22	30.55	3345	
16	30	600	800	732.87	199.27	9e-4	52.98	3093	1e-4	61.28	25	84.30	3761	
17	50	1200	100	51.91	56.21	1e-3	10.74	3580	1e-4	3.69	48	33.20	8985	
18	50	1200	200	224.47	101.93	1e-3	30.42	4666	1e-4	34.27	43	59.89	7536	
19	50	1200	400	833.57	227.48	9e-4	79.22	4499	1e-4	52.60	34	154.41	7630	
20	50	1200	800	3749.56	468.26	8e-4	180.41	4900	1e-4	76.22	25	168.72	4174	

Table 1 Comparison of the best SM with Cplex and Bundle methods

The results are reported in Table 1, which shows some interesting trends.

671 While for small-scale instances direct use of an LP solver is the best option, de-
672 composition approaches become more and more competitive as the size grows.
673 Often the Bundle method using “complex” master problems (DE-L) is the best
674 option; the approach also has the advantage that one can get very high-quality
675 dual solutions, and the corresponding accurate optimal primal solutions, with
676 a comparatively minor increase in effort [29]. However, as the size increases,
677 the master problem cost becomes very high; thus, methods that use cheaper
678 master problems can be competitive even if they require many more iterations.
679 In particular, with only one exception (group 20), KVC is faster than AK-Q,
680 while obtaining a roughly comparable gap; it is fair to remark, however, that
681 KVC did not always attain the required $1e-4$ accuracy, although it was always
682 pretty close, whereas AK-Q always did. Yet, this confirms previous experience
683 [13] that aggregated Bundle methods do not always attain significantly higher
684 convergence rates than well-tuned SM, despite collecting far more information
685 and paying the corresponding price in terms of master problem time. Inter-
686 estingly, in several cases (groups 2, 4–8, 10 and 12), KVC obtains comparable
687 gaps than DE-L in less time, often significantly so. These results requiring ac-
688 curate selection of the many parameters, and partly hinge on the availability
689 of (at least approximate) bounds on the optimal value of the problem; hence,
690 standard techniques like the use of general-purpose solvers, or even more stable
691 nondifferentiable optimization approaches like Bundle methods, can be more
692 appropriate if these conditions are not met. However, our study confirms that
693 appropriately tuned SM can be competitive for efficiently computing (not too
694 tight) bounds for hard, large-scale IPs.

695 4 Conclusion

696 We have computationally analysed a large class of Subgradient Methods, cov-
697 ering many of the ones proposed in the literature so far, for the solution of
698 large-scale Lagrangian Duals of hard Integer Programs. The specific features
699 of this application are that the number of variables is large, the computation
700 of the function decomposes into many independent problems, and only a rel-
701 atively poor accuracy is required. Our results show that, although the total
702 number of variants (comprised the possible settings for the numerical algorith-
703 mic parameters) is rather large, it is not exceedingly difficult to find settings
704 that work reasonably well across a large family of instances. Provided that
705 the appropriate tuning is made, SM perform roughly as expected: while their
706 global rate of convergence is far from being appealing, their very low cost per
707 iteration—in particular, outside of the function computation—can make up
708 for it as long as a relatively coarse bound is required.

709 Our interest in performing these experiments was partly about understand-
710 ing the computational significance of the theory developed in [21]. In this
711 sense, we can report that the ideas developed therein actually seem to have
712 an impact: deflection is indeed crucial for good performances of a SM, and
713 deflection and projection do work better together (cf. Table 2). Interestingly,

714 deflection-restricted approaches, developed for proving theoretical convergence
715 of SM, actually seem to work well in practice in some cases (cf. Table 3). What
716 mostly motivated our interest, however, was the hope that two relatively re-
717 cent additions to the arsenal of SM, namely incremental and primal-dual ap-
718 proaches, could significantly improve the performances with respect to more
719 “traditional” ones. Limited to the very specific instances and problems we have
720 tested, and against our expectations, this proved less successful. In hindsight,
721 this might have been expected for incremental methods: the size of the vari-
722 ables space is large, while the subproblems are of very low complexity, which
723 means that the “fixed cost” for each iteration (even if AS techniques are ap-
724 plied) largely makes partial computation of f irrelevant. There very likely are
725 IPs where these trade-offs are different, and therefore incremental methods can
726 be competitive, especially if theoretical developments—e.g., along the lines of
727 [63]—would allow incorporating deflection techniques. As far as PDSM are
728 concerned, the results are promising in that they show a very consistent be-
729 haviour with a much lower need of tuning parameters. Still, carefully tuned
730 version of traditional SM can significantly outperform them in most scenarios.
731 Our results seem to suggest that PDSM may be improved in practice by:

- 732 – exploiting information about the optimal value of the problem;
- 733 – adapting the approach to cope with an active-set strategy;
- 734 – adapting the theory to cope with cases where the feasible set, and even
735 worse the *optimal* set, is unbounded.

736 We hope that our analysis will stimulate further research along these lines.

737 A different line of research concerns the actual use of SM within enumer-
738 ative approaches for the IP. In such a framework, trading faster bound com-
739 putation for lower bound quality can indeed improve the overall efficiency of
740 the approach, but only if the right trade-offs are made. Furthermore, solution
741 of the LD is required not once, but in each B&C node; hence, *reoptimization*
742 techniques, whereby the information generated at the parent node is exploited
743 to improve the solution time at its descendants, become crucial. Which SM
744 are more efficient in this context, in terms of the global running time of the
745 enumerative algorithm rather than of any single bound computation, is an
746 open question that we intend to pursue in the future.

747 Acknowledgements

748 The first author acknowledge the contribution of the Italian Ministry for Uni-
749 versity and Research under the PRIN 2012 Project 2012JXB3YF “Mixed-
750 Integer Nonlinear Optimization: Approaches and Applications”. The work
751 of the second author has been supported by NSERC (Canada) under grant
752 184122-09. The work of the third author has been supported by the Post-
753 Doctoral Fellowship D.R. No 2718/201 (Regional Operative Program Calabria
754 ESF 2007/2013) and the Interuniversity Attraction Poles Programme P7/36
755 “COMEX: combinatorial optimization metaheuristics & exact methods” of
756 the Belgian Science Policy Office. All the authors gratefully acknowledge the

757 contribution of the anonymous referees and of the editors of the journal to
758 improving the initial version of the manuscript.

759 A Appendix

760 We now describe all the details of the SM that we have tested, together with the results of the
761 tuning phase. We remark that for some parameters it is nontrivial even to set a reasonable
762 ranges for the values. Our approach has been to select the initial range heuristically, and
763 then test it: if the best value consistently ended up being at one extreme, this was taken as
764 indication that the interval should be enlarged accordingly. This hinges on the assumption
765 that the behaviour of the algorithm is somewhat “monotonic” in the parameters; while this
766 is not necessarily true, for the vast majority of parameters a “monotonic” behaviour has
767 been verified experimentally, in that we almost never found the case where different settings
768 “far apart” provided better performances than these “in the middle.”

769 A.1 General parameters of SM

770 The following parameters are common to all variants of SM we tested, basically irrespective
771 of the specific rules for choosing the stepsize and the deflection.

- 772 – We denote by $\mathbf{pr} \subseteq \{g_i, d_{i-1}, d_i\}$ the subset of vectors that are projected on the
773 tangent cone T_i of Λ at $\bar{\lambda}_i$; in all our tests, \mathbf{pr} does not depend on the iteration. As
774 already remarked, $\mathbf{pr} = \{g_i, d_{i-1}, d_i\}$ makes no sense as T_i is convex. Furthermore,
775 when no deflection is done $d_i = g_i$ and therefore only $\mathbf{pr} = \{g_i\}$ and $\mathbf{pr} = \emptyset$ make sense.
- 776 – Regarding the order in which the stepsize and the deflection are chosen, we denote by
777 $\mathbf{sg} \in \{\text{drs}, \text{dr0}, \text{srs}, \text{sr0}\}$ the four possible schemes, where “dr” and “sr” refer to the
778 deflection-restricted and stepsize-restricted approach, respectively, while “s” and “0”
779 refer to using or not the safe rule ((10) and (9), respectively). Of course, drs and dr0
780 only apply if deflection is performed.
- 781 – We denote by χ the parameter used to adjust the Lipschitz constant L in the incremental
782 case, cf. (14), for which we tested the values $\chi = \mathbf{1e-v}$ for $v \in \{0, \dots, 8\}$.
- 783 – For the AS, one crucial decision is how often separation is performed: doing it less often
784 avoids some computations, but at the risk of ignoring possibly relevant information for
785 too long. We performed separation after the fixed number $s_1 \in \{0, 1\}$ of iterations, i.e.,
786 either not using the AS at all or separating every iteration. Initial tests showed that
787 larger values of s_1 were not effective.

788 A.2 Parameters of the SR

789 We now examine in details the parameters of the three SR. Since all of them have the form
790 (7), we are looking at different ways for determining β_i and f_i^{lev} .

791 **Polyak** In this SR β_i and f_i^{lev} are kept fixed at all iterations. Here, we exploit the fact
792 that in our application we know have “target value” \underline{f} and simply test the two cases $f^{lev} \in$
793 $\{f_*, 10\%f_*\}$. As for the other parameter, we tested $\beta \in \{0.01, 0.1, 1, 1.5, 1.99\}$.

794 **ColorTV** This SR is based on the improvement $\Delta f = \bar{f}_{i-1} - f_i$ of f and the scalar product
795 $d_i g_i$ to estimate “how successful a step has been.” Note, however, that in deflection-restricted
796 schemes (i.e., drs and dr0) d_i is not available and we use $d_{i-1} g_i$ instead. Iteration i is marked
797 as **green** if $d_i g_i > \rho$ and $\Delta f \geq \rho \max\{|f_i^{rec}|, 1\}$, as **yellow** if $d_i g_i < \rho$ and $\Delta f \geq 0$, and
798 as **red** otherwise, where $\rho > 0$ is a tolerance. Intuitively, **green** is a “good” step possibly
799 indicating that a larger ν_i may have been preferable, whereas **red** is a “bad” one suggesting
800 that ν_i is too large. Given three parameters c_g, c_y and c_r , and denoting by n_g, n_y and n_r
801 the number of *consecutive* **green**, **yellow** and **red** iterations, respectively, β_i is updated as:

- 802 1. if $n_g \geq c_g$ then set $\beta_i = \min\{2, 2\beta_{i-1}\}$;
- 803 2. if $n_y \geq c_y$ then set $\beta_i = \min\{2, 1.1\beta_{i-1}\}$;

- 804 3. if $n_r \geq c_r$ then then set $\beta_i = \max\{5e-4, 0.67\beta_{i-1}\}$;
 805 4. if none of the above cases occur, then set $\beta_i = \beta_{i-1}$.

806 One important parameter is therefore the arbitrarily fixed value β_0 . Also, the SR includes a
 807 simple target-following scheme whereby if $f_i \leq 1.05f_i^{lev}$ then $f_i^{lev} = f_i - 0.05f_i^{lev}$ (note that
 808 this never happens for $f^{lev} = 10\%f_*$). For this SR we kept $\rho = 1e-6$ fixed and we tested all
 809 combinations of $\beta_0 \in \{0.01, 0.1, 1, 1.5, 1.99\}$, $c_g \in \{1, 10, 50\}$, $c_y \in \{50, 100, 400\}$,
 810 and $c_r \in \{10, 20, 50\}$.

811 **FumeroTV** This SR has a complex management of f_i^{lev} and β_i , motivated by experimental
 812 considerations [31], that is subdivided into two distinct phases. The switch between the two
 813 is an iteration counter r , that is increased each time there is no improvement in the function
 814 value. This counter is used to define the exponential function $\sigma(r) = e^{-0.6933(r/r_1)^{3.26}}$,
 815 where r_1 is a parameter; note that $\sigma(r_1) \approx 1/2$, which is how the two apparently weird
 816 numerical parameters have been selected. The function σ , which is decreasing in r , is used
 817 in two ways. The first is to determine the maximum number of non-improving steps, which
 818 is the smallest integer r_2 such that $\sigma_\infty \geq \sigma(r_2)$, where the threshold $\sigma_\infty > 0$ is another
 819 parameter: given r_1 and σ_∞ , r_2 can be obtained with a simple closed formula. The second
 820 is to construct at each iteration the value of f_i^{lev} as a convex combination of the known
 821 global lower bound \underline{f} (which, not incidentally, this algorithm specifically tailored for IP
 822 is the only one to explicitly use) and the current record value as $f_i^{lev} = \sigma(r)\underline{f} + (1 -$
 823 $\sigma(r))f_i^{rec}$. In the first phase, when r varies, the threshold varies as well: as $\sigma(r)$ decreases
 824 when r grows, f_i^{lev} is kept closer and closer to f_i^{rec} as the algorithm proceeds. In the second
 825 phase ($r \geq r_2$), where r is no longer updated, $\sigma(r) = \sigma_\infty$. The procedure for updating
 826 r and β_i uses four algorithmic parameters: a tolerance $\delta > 0$, two integer numbers η_1
 827 and $\eta_2 \geq 1$, and the initial value $\beta_0 \in (0, 2)$. The procedure is divided in two phases,
 828 according to the fact that the iteration counter r (initialized to 0) is smaller or larger than
 829 the threshold r_2 . Similarly to **ColorTV**, the rule keeps a record value \bar{f}_i (similar, but not
 830 necessarily identical, to f_i^{rec}) and declares a “good” step whenever $f_i \leq \bar{f}_i - \delta \max\{|\bar{f}_i|, 1\}$,
 831 in which case \bar{f} is updated to f_i . In either phase, the number of consecutive “non-good”
 832 steps is counted. In the first phase, after $\bar{\eta}_2$ such steps r is increased by one, and β_i is
 833 updated as $\beta_i = \beta_{i-1}/(2\beta_{i-1} + 1)$. In the second phase r is no longer updated: after
 834 every “good” step β_i is doubled, whereas after $\bar{\eta}_1$ “non good” steps β_i is halved. In the
 835 tuning phase we tested the following values for the parameters: $\sigma_\infty \in \{1e-4, 1e-3, 1e-2$
 836 $\}$, $\delta = 1e-6$, $r_1 \in \{10, 50, 100, 150, 200, 250, 300, 350\}$, $\beta_0 \in \{0.01, 0.1, 1, 1.5, 1.99\}$,
 837 $\eta_1 \in \{10, 50, 100, 150, 200, 250, 300, 350\}$, $\eta_2 \in \{10, 50, 100, 150, 200\}$.

838 A.3 Parameters of the DR

839 We now describe in details the two “complex” DR that we have tested (**STSubgrad**, where
 840 $\alpha_i = 1 \implies d_i = g_i$ and $\bar{\lambda}_{i+1} = \lambda_{i+1}$ for all i , hardly needs any comment). Note that the
 841 selection of $\bar{\lambda}_{i+1}$ is also done by the **Deflection()** object.

842 **Primal-Dual** The PDSM is based on a sophisticated convergence analysis aimed at ob-
 843 taining optimal a-priori complexity estimates [54]. A basic assumption of PDSM is that A
 844 is endowed with a *prox-function* $d(\lambda)$, and that one solves the modified form of (1)

$$\min\{f(\lambda) : d(\lambda) \leq D, \lambda \in A\} \quad (21)$$

845 restricted upon a compact subset of the feasible region, where $D \geq 0$ is a parameter. D is
 846 never directly used in the algorithm, except to optimally tune its parameters; hence, (21)
 847 can always be considered if f has a minimum λ_* . In particular, we take $d(\lambda) = \|\lambda - \lambda_0\|^2/2$,
 848 in which case $D = \|\lambda_* - \lambda_0\|^2/2$. In general D is unknown; however, the parameter “ t^* ” in
 849 the stopping formulæ (12)/(13) is somehow related. Roughly speaking, t^* estimates how far
 850 at most one can move along a subgradient $g_i \in \partial f(\lambda_i)$ when λ_i is an approximately optimal
 851 solution. The parameter, that it used in the same way by Bundle methods, is independent
 852 from the specific solution algorithm and has been individually tuned (which is simple enough,
 853 ex-post); hence, $D = (t^*)^2 L$ is a possible estimate. Yet, t^* is supposed to measure $\|\lambda_* - \lambda_i\|$
 854 for a “good” λ_i , whereas D requires the initial λ_0 , which typically is not “good”: hence,

we introduced a further scaling factor $F > 0$, i.e., took $\gamma = (F\sqrt{L})/(t^*\sqrt{2})$ for SA and $\gamma = F/(t^*\sqrt{2L})$ for WA (cf. (25)), and we experimentally tuned F . In general one would expect $F > 1$, and the results confirm this; however, to be on the safe side we tested all the values $F \in \{1e-4, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4\}$. As suggested by one Referee we also tested using $D = \|\lambda_* - \lambda_0\|^2/2$, with λ_* obtained by some previous optimization. The results clearly showed that the “exact” estimate of D not always translated in the best performances; in particular, for the FR the results were always consistently worse, whereas for the KR the results were much worse for WA, and completely comparable (but not any better) for SA. This is why in the end we reported results with the tuned value of F .

For the rest, PDSM basically have no tunable parameters. It has to be remarked, however, that PDSM are not, on the outset, based on a simple recurrence of the form (5); rather, given two sequences of weights $\{v_i\}$ and $\{\omega_i\}$, the next iterate is obtained as

$$\lambda_{i+1} = \operatorname{argmin}\left\{\lambda \sum_{k=1}^i v_k g_k + \omega_i d(\lambda) : \lambda \in \Lambda\right\} . \quad (22)$$

Yet, when $\Lambda = \mathbb{R}^n$ (22) readily reduces to (5), as the following Lemma shows.

Lemma 1 *Assume $\Lambda = \mathbb{R}^n$, select $d(\lambda) = \|\lambda - \lambda_0\|^2/2$, fix $\lambda_i = \lambda_0$ for all $i \geq 0$ in (5). By defining $\Delta_i = \sum_{k=1}^i v_k$, the following DR and SR*

$$\alpha_i = v_i/\Delta_i \quad (\in [0, 1]) \quad \text{and} \quad \nu_i = \Delta_i/\omega_i \quad (23)$$

are such that λ_{i+1} produced by (22) is the same produced by (5) and (8).

Proof Under the assumptions, (22) is a strictly convex unconstrained quadratic problem, whose optimal solution is immediately available by the closed formula

$$\lambda_{i+1} = \lambda_0 - (1/\omega_i) \sum_{k=1}^i v_k g_k . \quad (24)$$

This clearly is (5) under the SR in (23) provided that one shows that the DR in (23) produces

$$d_i = (\sum_{k=1}^i v_k g_k)/\Delta_i .$$

This is indeed easy to show by induction. For $i = 1$ one immediately obtains $d_1 = g_1$. For the inductive case, one just has to note that

$$1 - \frac{v_{i+1}}{\Delta_{i+1}} = \frac{\Delta_{i+1} - v_{i+1}}{\Delta_{i+1}} = \frac{\Delta_i}{\Delta_{i+1}}$$

to obtain

$$d_{i+1} = \alpha_{i+1} g_{i+1} + (1 - \alpha_{i+1}) d_i = \frac{v_{i+1}}{\Delta_{i+1}} g_{i+1} + \frac{\Delta_i}{\Delta_{i+1}} \frac{\sum_{k=1}^i v_k g_k}{\Delta_i} = \frac{1}{\Delta_{i+1}} \sum_{k=1}^{i+1} v_k g_k . \quad \square$$

Interestingly, the same happens if simple sign constraints $\lambda \geq 0$ are present, which is what we actually have whenever $\Lambda \neq \mathbb{R}^n$.

Lemma 2 *If $\Lambda = \mathbb{R}_+^n$, the same conclusion as in Lemma 1 hold after $P_\Lambda(\lambda_{i+1})$.*

Proof It is easy to see that the optimal solution of (22) with $\Lambda = \mathbb{R}_+^n$ is equal to that with $\Lambda = \mathbb{R}^n$, i.e. (24), projected over \mathbb{R}_+^n . \square

Therefore, implementing the DR and the SR as in (23), and *never* updating $\bar{\lambda}_i = \lambda_0$, allow us to fit PDSM in our general scheme. To choose v_i and ω_i we follow the suggestions in [54]: the SA approach corresponds to $v_i = 1$, and the WA one to $v_i = 1/\|g_i\|$. We then set $\omega_i = \gamma \hat{\omega}_i$, where $\gamma > 0$ is a constant, and $\hat{\omega}_0 = \hat{\omega}_1 = 1$, $\hat{\omega}_i = \hat{\omega}_{i-1} + 1/\hat{\omega}_{i-1}$ for $i \geq 2$, which implies $\hat{\omega}_{i+1} = \sum_{k=0}^i 1/\hat{\omega}_k$. The analysis in [54] suggests settings for γ that provide the best possible theoretical convergence, i.e.,

$$\gamma = L/\sqrt{2D} \quad \text{and} \quad \gamma = 1/\sqrt{2D} , \quad (25)$$

for the SA and WA, respectively, L being the Lipschitz constant of f .

887 **Volume** In this DR, α_i is obtained as the optimal solution of a univariate quadratic problem. As suggested in [5], and somewhat differently from the original [6], we use exactly the
 888 “poorman’s form” of the master problem of the proximal Bundle method
 889

$$\min \{ \nu_{i-1} \|\alpha g_i + (1 - \alpha)d_{i-1}\|^2 / 2 + \alpha \sigma_i(\bar{\lambda}_i) + (1 - \alpha)\epsilon_{i-1}(\bar{\lambda}_i) : \alpha \in [0, 1] \} \quad (26)$$

890 where the linearization errors $\sigma_i(\bar{\lambda}_i)$ and $\epsilon_{i-1}(\bar{\lambda}_i)$ have been discussed in details in §2.2.
 891 Note that we use the stepsize ν_{i-1} of the previous iteration as *stability weight*, since that
 892 term corresponds to the stepsize that one would do along the dual optimal solution in a
 893 Bundle method [3, 5, 25]. It may be worth remarking that the dual of (26)

$$\min \{ \max\{g_i d - \sigma_i(\bar{\lambda}_i), d_{i-1} d - \epsilon_{i-1}(\bar{\lambda}_i)\} + \|d\|^2 / (2\nu_{i-1}) \} , \quad (27)$$

894 where $d = \lambda - \bar{\lambda}_i$, is closely tied to (22) in PDSM. The difference is that (27) uses two (approximate) subgradients, g_i and d_i , whereas in (22) one uses only one (approximate) subgradient
 895 obtained as weighted average of the ones generated at previous iterations. Problem (26) is
 896 inexpensive, because without the constraint $\alpha \in [0, 1]$ it has the closed-form solution
 897

$$\alpha_i^* = \frac{\epsilon_{i-1}(\bar{\lambda}_i) - \sigma_i(\bar{\lambda}_i) - \nu_{i-1} d_{i-1} (g_i - d_{i-1})}{\nu_{i-1} \|g_i - d_{i-1}\|^2} ,$$

898 and thus one can obtain its optimal solution by simply projecting α_i^* over $[0, 1]$. However,
 899 as suggested in [5, 6] we rather chose α_i in the more safeguarded way

$$\alpha_i = \begin{cases} \alpha_{i-1}/10 & \text{if } \alpha_i^* \leq \mathbf{1e-8} \\ \min\{\tau_i, 1.0\} & \text{if } \alpha_i^* \geq 1 \\ \alpha_i^* & \text{otherwise} \end{cases}$$

900 where τ_i is initialized to τ_0 , and each τ_p iterations is decreased multiplying it by $\tau_f <$
 901 1, while ensuring that it remains larger than τ_{\min} . The choice of the stability center is
 902 also dictated by a parameter $m > 0$ akin that used in Bundle methods: if $f_i - f_{i+1} \geq$
 903 $m \max\{1, |f_i^{ref}|\}$ a Serious Step occurs and $\bar{\lambda}_{i+1} = \lambda_{i+1}$, otherwise a Null Step takes place
 904 and $\bar{\lambda}_{i+1} = \bar{\lambda}_i$. For the tuning phase we have searched all the combinations of the following
 905 values for the above parameters: $\tau_0 \in \{0.01, 0.1, 1, 10\}$, $\tau_p \in \{10, 50, 100, 200, 500\}$,
 906 $\tau_f \in \{0.1, 0.4, 0.8, 0.9, 0.99\}$, $\tau_{\min} \in \{\mathbf{1e-4}, \mathbf{1e-5}\}$, $m \in \{0.01, 0.1\}$.

907 A.4 Detailed results of the tuning phase

908 The tuning phase required a substantial computational work, and a nontrivial analysis of
 909 the results. As discussed in §3.2, each SM configuration gave rise to an aggregated conver-
 910 gence graph. To select the best configurations, the graphs were visually inspected, and the
 911 ones corresponding to a better overall convergence rates were selected. This usually was the
 912 configuration providing the best final gap for all instances. Occasionally, other configura-
 913 tions gave better results than the chosen one in the earlier stages of the algorithm on some
 914 subsets of the instances; usually the advantage was marginal at best, and only on a fraction
 915 of the cases, while the disadvantage in terms of final result was pronounced. In general it
 916 has always been possible to find “robust” settings that provided the best (or close so) gap
 917 at termination, but were not too far from the best gaps even in all the other stages. Further-
 918 more, although the total number of possible combinations was rather large, it turned out
 919 that only a relatively small set of parameters had a significant impact on the performances,
 920 and in most of the cases their effect was almost orthogonal to each other. This allowed us
 921 to effectively single out “robust” configurations for our test sets; for several of the param-
 922 eters, the “optimal” choice has been unique across all instances, which may provide useful
 923 indications even for different problems.

924 For the sake of clarity and conciseness, in Tables 2 and 3, we report the chosen values
 925 of the parameters for FR and KR, respectively, briefly remarking about the effect of each
 926 parameter and their relationships. The behaviour of SM was pretty similar in the two cases
 927 $\underline{f} = f_*$ and $\underline{f} = 10\%f_*$; hence, the tables report the values for $\underline{f} = f_*$, indicating in “[]” these
 928 for $\underline{f} = 10\%f_*$ if they happen to be different. The tables focus on the combinations between
 929 the three SR and the two DR, plus the incremental case; the parameters of **Primal-Dual**
 930 variant are presented separately since the SR is combined with the DR.

931 *Results for the FR.* The results for FR are summarized in Table 2, except for those
 932 settings that are constantly optimal. In particular, **STSubgrad** and **Incremental** have better
 933 performances with $\text{pr} = \{g_i\}$, irrespective of the SR. For **Volume**, instead, the optimal setting
 934 of pr does depend on the SR, although $\text{pr} = \{d_i\}$ and $\text{pr} = \{d_{i-1}\}$ were hardly different. All
 935 the other parameters of **Volume** depend on the SR (although the stepsize-restricted scheme
 936 with no safe rule is often good), except τ_{\min} and m that are always best set to $1e-4$ and
 937 0.1 , respectively. Another interesting observation is that, while **Volume** does have several
 938 parameters, it does seem that they operate quite independently of each other, as changing
 939 one of them always has a similar effect irrespective of the others. We also mention that for
 940 **ColorTV** the parameters c_y and c_r have little impact on the performance, whereas c_g plays
 941 an important role and it significantly influences the quality of the results. As for **FumeroTV**,
 942 σ_∞ and η_2 have hardly any impact, and we arbitrarily set them to $1e-4$ and 50 , respectively.

	Polyak	ColorTV				FumeroTV		
	β_i	β_0	c_g	c_y	c_r	β_0	r_1	η_1
Volume	0.01	0.1	50	400	50	0.1	150	50
					[10]		[100]	
τ_0	10			1			1	
τ_f	.8			.8			.8 [9]	
τ_i	200 [100]			100			100 [200]	
pr	$\{d_i, d_{i-1}\} [d_{i-1}]$			g_i			$g_i [d_{i-1}]$	
sg	$sr0 [srs]$			$sr0$			$sr0 [drs]$	
STSubgrad	1.5	1.5	1	50	50	1.99	200	250
		[0.01]				[1.5]	[50]	[150]
sg	$sr0$			$sr0 [srs]$			$sr0$	
Incremental	1.5	1.99	50	100	50	1.99	300	300
	[0.1]	[1.5]	[10]	[400]		[1.5]	[50]	[100]
χ	$1e-3 [1e-2]$			$1e-3$			$1e-3$	
sg	$sr0 [srs]$			$sr0 [srs]$			$sr0$	

Table 2 Optimal parameters for the *Flow Relaxation*

943 In PDSM, the only crucial value is F , used to compute the optimal value of γ in (25). We
 944 found its best value to be $1e2$ and $1e3$ for SA and WA, respectively. The choice has a large
 945 impact on performances, which significantly worsen for values far from these.

946 *Results for the KR.* The best parameters for the KR are reported in Table 3. Although
 947 the best values are in general different from the FR, confirming the (unfortunate) need for
 948 problem-specific parameter tuning, similar observations as in that case can be made. For
 949 instance, for **Volume**, the parameters were still more or less independent from each other, and
 950 τ_{\min} and m were still hardly impacting, with the values $1e-4$ and 0.1 still very adequate. For
 951 **ColorTV**, results are again quite stable varying c_y . Yet, differences can be noted: for instance,
 952 for FR c_g is clearly the most significant parameter and dictates most of the performance
 953 variations, while for the KR the relationship between the two parameters c_r and c_g and the
 954 results is less clear. Similarly, for **FumeroTV** some settings are conserved: σ_∞ and η_2 have
 955 very little effect and can be set to $1e-4$ and 50 , respectively. Other cases were different: for
 956 instance the parameters η_1 , r_1 and β_0 were more independent on each other than in the FR.

957 The parameters of **Primal-Dual** showed to be quite independent from the underlying
 958 Lagrangian approach, with the best value of F still being $1e2$ for SA and $1e3$ for WA. This
 959 confirms the higher overall robustness of the approach.

960 We terminate the Appendix with a short table detailing which of the variants of SM
 961 that we tested have a formal proof of convergence and where it can be found, indicating the

		Polyak	ColorTV				FumeroTV		
		β_i	β_0	c_g	c_y	c_r	β_0	r_1	η_1
		0.1	0.1	50	50	50	0.1	10	10
						[10]	[50]	[50]	
Volume	τ_0	1	1				1[10]		
	τ_f	.9 [.8]	.8 [.9]				.99 [.8]		
	τ_i	50	100 [50]				50 [200]		
	sg	$dr0$ [srs]	$dr0$ [srs]				$dr0$ [drs]		
STSubgrad		1.5 [.1]	.01	50	50	50	1.99	50	250
						[10]	[200]		
sg		$sr0$	$sr0$				$sr0$		
Incremental		1.5 [.1]	1 [.01]	50 [10]	100	50	1.5 [1]	100 [10]	100
χ		$1e-5$ [$1e-6$]	$1e-5$				$1e-5$		
sg		$sr0$	srs				$sr0$		

Table 3 Optimal parameters for the *Knapsack Relaxation*

962 references wherein the proofs are given. The columns DR and SR, as usual, indicate which
 963 ones among the possible defection and stepsize rules are adopted; an entry “any” means
 964 that the corresponding proof holds for all the rules. Moreover, PR, AS and IN, respectively,
 965 stands for the strategies: (i) projection, (ii) active set and (iii) incremental.

DR	SR	PR	AS	IN	Reference
Primal-Dual		no	no	no	[55]
STSubgrad	Polyak	no	no	no	[58]
STSubgrad	FumeroTV	no	no	no	[31]
STSubgrad	any	yes	no	no	[21]
STSubgrad	any	no	no	yes	[9]
Volume	ColorTV	no	no	no	[6]
Volume	any	yes	no	no	[21]

Table 4 Theoretical convergence proofs of the employed SM

966 References

- 967 1. Ahookhosh, M.: Optimal subgradient algorithms with application to large-scale linear
 968 inverse problems. Tech. rep., Optimization Online (2014)
- 969 2. Anstreicher, K., Wolsey, L.: Two “well-known” properties of subgradient optimization.
 970 Mathematical Programming **120**(1), 213–220 (2009)
- 971 3. Astorino, A., Frangioni, A., Fuduli, A., Gorgone, E.: A nonmonotone proximal bundle
 972 method with (potentially) continuous step decisions. SIAM Journal on Optimization
 973 **23**(3), 1784–1809 (2013)
- 974 4. Bacaud, L., Lemaréchal, C., Renaud, A., Sagastizábal, C.: Bundle methods in stochastic
 975 optimal power management: A disaggregated approach using preconditioners. Compu-
 976 tational Optimization and Applications **20**, 227–244 (2001)
- 977 5. Bahiense, L., Maculan, N., Sagastizábal, C.: The volume algorithm revisited: Relation
 978 with bundle methods. Mathematical Programming **94**(1), 41–70 (2002)

- 979 6. Barahona, F., Anbil, R.: The volume algorithm: Producing primal solutions with a
980 subgradient method. *Mathematical Programming* **87**(3), 385–399 (2000)
- 981 7. Beck, A., Teboulle, M.: Smoothing and first order methods: a unified framework. *SIAM*
982 *Journal on Optimization* **22**(2), 557–580 (2012)
- 983 8. Ben Amor, H., Desrosiers, J., Frangioni, A.: On the choice of explicit stabilizing terms
984 in column generation. *Discrete Applied Mathematics* **157**(6), 1167–1184 (2009)
- 985 9. Bertsekas, D., Nedić, A.: Incremental subgradient methods for nondifferentiable opti-
986 mization. *SIAM J. on Optimization* **12**(1), 109–138 (2001)
- 987 10. Borghetti, A., Frangioni, A., Lacalandra, F., Nucci, C.: Lagrangian heuristics based on
988 disaggregated bundle methods for hydrothermal unit commitment. *IEEE Transactions*
989 *on Power Systems* **18**(1), 313–323 (February 2003)
- 990 11. Bot, R., Hendrich, C.: A variable smoothing algorithm for solving convex optimization
991 problems. *TOP* (2014)
- 992 12. Brännlund, U.: A generalised subgradient method with relaxation step. *Mathematical*
993 *Programming* **71**, 207–219 (1995)
- 994 13. Briant, O., Lemaréchal, C., Meurdesoif, P., Michel, S., Perrot, N., Vanderbeck, F.: Com-
995 parison of bundle and classical column generation. *Mathematical Programming* **113**(2),
996 299–344 (2008)
- 997 14. Camerini, P., Fratta, L., Maffioli, F.: On improving relaxation methods by modified
998 gradient techniques. *Mathematical Programming Study* **3**, 26–34 (1975)
- 999 15. Cappanera, P., Frangioni, A.: Symmetric and asymmetric parallelization of a cost-
1000 decomposition algorithm for multi-commodity flow problems. *INFORMS Journal on*
1001 *Computing* **15**(4), 369–384 (2003)
- 1002 16. Censor, Y., Davidi, R., Herman, G., Schulte, R., Tretuashvili, L.: Projected subgradient
1003 minimization cersus superiorization. *Journal on Optimization Theory and Applications*
1004 **160**(3), 730–747 (2014)
- 1005 17. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with
1006 applications to imaging. *Journal of Mathematical Imaging and Vision* **40**(1), 120–145
1007 (2011)
- 1008 18. Crainic, T., Frangioni, A., Gendron, B.: Multicommodity capacitated network design.
1009 *Telecommunications Network Planning* pp. 1–19 (1999)
- 1010 19. Crainic, T., Frangioni, A., Gendron, B.: Bundle-based relaxation methods for multi-
1011 commodity capacitated fixed charge network design problems. *Discrete Applied Math-*
1012 *ematics* **112**, 73–99 (2001)
- 1013 20. Crema, A., Loreto, M., Raydan, M.: Spectral projected subgradient with a momen-
1014 tum term for the Lagrangean dual approach. *Computers & Operations Research* **34**,
1015 31743186 (2007)
- 1016 21. d’Antonio, G., Frangioni, A.: Convergence analysis of deflected conditional approximate
1017 subgradient methods. *SIAM Journal on Optimization* **20**(1), 357–386 (2009)
- 1018 22. du Merle, O., Goffin, J.L., Vial, J.P.: On improvements to the analytic center cutting
1019 plane method. *Computational Optimization and Applications* **11**, 37–52 (1998)
- 1020 23. Feltenmark, S., Kiwiel, K.: Dual applications of proximal bundle methods, including
1021 Lagrangian relaxation of nonconvex problems. *SIAM Journal on Optimization* **10**(3),
1022 697–721 (2000)
- 1023 24. Frangioni, A.: Solving semidefinite quadratic problems within nonsmooth optimization
1024 algorithms. *Computers & Operations Research* **21**, 1099–1118 (1996)
- 1025 25. Frangioni, A.: Generalized bundle methods. *SIAM Journal on Optimization* **13**(1),
1026 117–156 (2002)
- 1027 26. Frangioni, A., Gallo, G.: A bundle type dual-ascent approach to linear multicommodity
1028 min cost flow problems. *INFORMS Journal on Computing* **11**(4), 370–393 (1999)
- 1029 27. Frangioni, A., Gendron, B.: A stabilized structured Dantzig-Wolfe decomposition
1030 method. *Mathematical Programming* **140**, 45–76 (2013)
- 1031 28. Frangioni, A., Gorgone, E.: A library for continuous convex separable quadratic knap-
1032 sack problems. *European Journal of Operational Research* **229**(1), 37–40 (2013)
- 1033 29. Frangioni, A., Gorgone, E.: Generalized bundle methods for sum-functions with “easy”
1034 components: Applications to multicommodity network design. *Mathematical Program-*
1035 *ming* **145**(1), 133–161 (2014)
- 1036 30. Frangioni, A., Lodi, A., Rinaldi, G.: New approaches for optimizing over the semimetric
1037 polytope. *Mathematical Programming* **104**(2-3), 375–388 (2005)

- 1038 31. Fumero, F.: A modified subgradient algorithm for Lagrangean relaxation. *Computers*
1039 *and Operations Research* **28**(1), 33–52 (2001)
- 1040 32. Geoffrion, A.: Lagrangian relaxation and its uses in iteger programming. *Mathematical*
1041 *Programming Study* **2**, 82–114 (1974)
- 1042 33. Gondzio, J., González-Brevis, P., Munari, P.: New developments in the primal-dual
1043 column generation technique. *European Journal of Operational Research* **224**(1), 41–
1044 51 (2013)
- 1045 34. Görtz, S., Klose, A.: A simple but usually fast branch-and-bound algorithm for the
1046 capacitated facility location problem. *INFORMS Journal on Computing* **24**(4), 597610
1047 (2012)
- 1048 35. Guignard, M.: Efficient cuts in Lagrangean ‘relax-and-cut’ schemes. *European Journal*
1049 *of Operational Research* **105**, 216–223 (1998)
- 1050 36. Held, M., Karp, R.: The traveling salesman problem and minimum spanning trees.
1051 *Operations Research* **18**, 1138–1162 (1970)
- 1052 37. Hiriart-Urruty, J.B., Lemaréchal, C.: *Convex Analysis and Minimization Algorithms*
1053 *II—Advanced Theory and Bundle Methods, Grundlehren Math. Wiss.*, vol. 306.
1054 Springer-Verlag, New York (1993)
- 1055 38. Ito, M., Fukuda, M.: A family of subgradient-based methods for convex optimization
1056 problems in a unifying framework. Tech. rep., Optimization Online (2014)
- 1057 39. Jones, K., Lustig, I., Farwolden, J., Powell, W.: Multicommodity network flows: the
1058 impact of formulation on decomposition. *Mathematical Programming* **62**, 95–117 (1993)
- 1059 40. Kelley, J.: The cutting-plane method for solving convex programs. *Journal of the SIAM*
1060 **8**, 703–712 (1960)
- 1061 41. Kiwiel, K.: Convergence of approximate and incremental subgradient methods for con-
1062 vex optimization. *SIAM Journal on Optimization* **14**(3), 807–840 (2003)
- 1063 42. Kiwiel, K., Goffin, J.: Convergence of a simple subgradient level method. *Mathematical*
1064 *Programming* **85**(4), 207–211 (1999)
- 1065 43. Kiwiel, K., Larsson, T., Lindberg, P.: The efficiency of ballstep subgradient level meth-
1066 ods for convex optimization. *Mathematics of Operation Research* **23**, 237–254 (1999)
- 1067 44. Lan, G., Zhou, Y.: Approximation accuracy, gradient methods, and error bound for
1068 structured convex optimization. Technical report, University of Florida (2014)
- 1069 45. Larsson, T., Patriksson, M., Strömberg, A.B.: Conditional subgradient optimization –
1070 theory and applications. *European Journal of Operational Research* **88**(2), 382–403
1071 (1996)
- 1072 46. Larsson, T., Patriksson, M., Strömberg, A.B.: Ergodic, primal convergence in dual sub-
1073 gradient schemes for convex programming. *Mathematical Programming* **86**, 283–312
1074 (1999)
- 1075 47. Lemaréchal, C.: An extension of Davidon methods to nondifferentiable problems. In:
1076 M. Balinski, P. Wolfe (eds.) *Nondifferentiable optimization, Mathematical Programming*
1077 *Study*, vol. 3, pp. 95–109. North-Holland, Amsterdam (1975)
- 1078 48. Lemaréchal, C., Renaud, A.: A geometric study of duality gaps, with applications.
1079 *Mathematical Programming* **90**, 399–427 (2001)
- 1080 49. Necoara, I., Suykens, J.: Application of a smoothing technique to decomposition in con-
1081 vex optimization. *IEEE Transactions on Automatic Control* **53**(11), 2674–2679 (2008)
- 1082 50. Nedic, A., Bertsekas, D.: Incremental subgradient methods for nondifferentiable opti-
1083 mization. *Mathematical Programming* **120**, 221–259 (2009)
- 1084 51. Nemirovski, A., Yudin, D.: *Problem Complexity and Method Efficiency in Optimization*.
1085 Wiley (1983)
- 1086 52. Nesterov, Y.: Excessive gap technique in nonsmooth convex minimization. *SIAM Jour-
1087 nal on Optimization* **16**, 235–249 (2005)
- 1088 53. Nesterov, Y.: Smooth minimization of non-smooth functions. *Mathematical Program-
1089 ming* **103**, 127–152 (2005)
- 1090 54. Nesterov, Y.: Primal-dual subgradient methods for convex optimization. *Mathematical*
1091 *Programming* **120**, 221–259 (2009)
- 1092 55. Nesterov, Y.: Universal gradient methods for convex optimization problems. *Mathe-
1093 matical Programming* (2014)
- 1094 56. Neto, E., De Pierro, A.: Incremental subgradients for constrained convex optimization:
1095 A unified framework and new methods. *SIAM Journal on Optimization* **20**(3), 1547–
1096 1572 (2009)

-
- 1097 57. Ouorou, A.: A proximal cutting plane method using Chebychev center for nonsmooth
1098 convex optimization. *Mathematical Programming* **119**(2), 239–271 (2009)
- 1099 58. Polyak, B.: Minimization of unsmooth functionals. *Zh.Vychisl.Mat.Fiz.* **9**(3), 509–521
1100 (1969)
- 1101 59. Serali, B., Choi, B., Tuncbilek, C.: A variable target value method for nondifferentiable
1102 optimization. *Operations Research Letters* **26**, 1–8 (2000)
- 1103 60. Serali, B., Lim, C.: On embedding the volume algorithm in a variable target value
1104 method. *Operations Research Letters* **32**, 455–462 (2004)
- 1105 61. Shor, N.: *Minimization Methods for Nondifferentiable Functions*. Springer-Verlag, Berlin
1106 (1985)
- 1107 62. Solodov, M., Zavriev, S.: Error stability properties of generalized gradient-type algo-
1108 rithms. *Journal of Optimization Theory and Applications* **98**(3), 663–680 (1998)
- 1109 63. Tseng, P.: Conditional gradient sliding for convex optimization. *Mathematical Pro-*
1110 *gramming* **125**, 263–295 (2010)
- 1111 64. Wolfe, P.: A method of conjugate subgradients for minimizing nondifferentiable func-
1112 tions. In: M. Balinski, P. Wolfe (eds.) *Nondifferentiable optimization*, *Mathematical*
1113 *Programming Study*, vol. 3, pp. 145–173. North-Holland, Amsterdam (1975)