Part I: Bundle Methods

I.1 A View of Bundle Algorithms

During the last 20 years, Bundle methods for Nondifferentiable Optimization (NDO) have become an important research area: stemming from the pioneering work of Lemaréchal, a large number of papers and books devoted to several variants of Bundle methods have appeared, both for unconstrained and constrained optimization. Nowadays, Bundle methods have a solid theoretical background as well as important applications (one for all, the solution of Lagrangean Duals): the main aim of all Part I is to present an application-oriented development of Bundle methods, with a convergence proof for a general form of the algorithm that explicates the very basic properties needed for the convergence analysis, thus allowing greater flexibility in the design of practical algorithms. When Bundle methods are applied to the optimization of Lagrangean functions, a "primal" interpretation¶ can be given to their basic ideas and algorithmic tools: such an interpretation leads to the development of a new class of algorithms with Linear Programming direction finding subproblems, which can be viewed as "stabilizations" of the Dantzig-Wolfe method, and it also suggests some new ideas for the most delicate task in any Bundle algorithm, i.e. the handling of the "trust region parameter". This Chapter is devoted to the presentation of the relevant terminology and ideas, together with an applications-oriented review of the Bundle approaches proposed in the literature: in particular, §I.1.1 deals with the "dual" viewpoint that is usually assumed in NDO theory, i.e. the maximization of a NonDifferentiable function provided as a "black box", while §I.1.2 illustrates the "primal" viewpoint.

I.1.1 The Dual viewpoint

Except for the treatment of the stopping criteria at the end, the material presented in this paragraph is not new: however, an attempt has been done to resume the large body of knowledge about Bundle methods in an original way. Bundle methods are perhaps the most important class of implementable algorithms for the numerical solution of the problem (that, for reasons to become clear in §I.1.2, is considered of a "dual" nature)

(D) $\max\{\ \varphi(\boldsymbol{y}) : \boldsymbol{y} \in Y\ \}$

where $\varphi()$ is a *concave* and *finite everywhere* (nondifferentiable) function and Y is a *convex nonempty* subset of $\mathcal{R}^n$. In the following development, we will adopt the usual assumption that $\varphi()$ be given as a "black box", which inputs a vector $\bar{\boldsymbol{y}} \in Y$ and outputs the value of $\varphi(\bar{\boldsymbol{y}})$ and one *subgradient*

¶ In a different sense from the one usually found in textbooks, e.g. [HL93 Chapter XV].

$\boldsymbol{g} = \boldsymbol{g}(\bar{\boldsymbol{y}})$ of $\varphi()$ in $\bar{\boldsymbol{y}}$, i.e. a vector of $\mathcal{R}^m$ that satisfies the *subgradient inequality*

$$\varphi(\boldsymbol{y}) \le \varphi(\bar{\boldsymbol{y}}) + \boldsymbol{g}(\boldsymbol{y} - \bar{\boldsymbol{y}}) \qquad \forall\ \boldsymbol{y}.$$

It should be noted that the definition of subgradient is usually given for *convex* functions, and is therefore slightly different from the above (which is that of a *supergradient* in the standard terminology); analogously, concepts like the *epigraph* of $\varphi()$

$$Epi(\varphi) = \{ (\boldsymbol{y}, v) \in \mathcal{R}^{m+1} : v \le \varphi(\boldsymbol{y}) \},$$

that are defined here for convenience in the setting of concave functions, may be found defined with a reversed inequality sign in a Convex Analysis text.

Geometrically, a subgradient is the gradient of a *supporting hyperplane* of $Epi(\varphi)$ in $\bar{\boldsymbol{y}}$ (Figure 1): unlike the gradient of a differentiable function, however, subgradients are not guaranteed to be directions of ascent (Figure 2). Although apparently poor, this first-order information is enough to drive convergent algorithms for the solution of (D): from the subgradient inequality, it easily follows that the level set $\{ \boldsymbol{y} : \varphi(\boldsymbol{y}) \ge \varphi(\bar{\boldsymbol{y}}) \}$, and therefore the set of the optimal solutions of (D), lies in the half-space pointed by $\boldsymbol{g}$ w.r.t. $\bar{\boldsymbol{y}}$ (Figure 2).

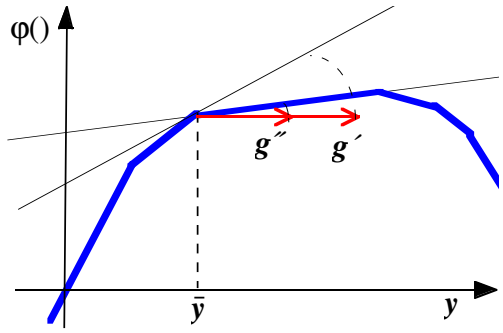


Figure 1: two subgradients $\boldsymbol{g}'$ and $\boldsymbol{g}''$ in a kink point

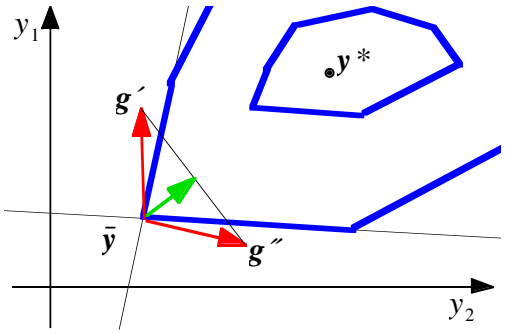


Figure 2: contour plots of $\varphi()$ and an optimal solution $\boldsymbol{y}^*$

It is well-known that any local optima of (D) is also a global optima, and that the problem enjoys many useful properties, such as continuity of $\varphi()$ on the relative interior of Y and existence of a subgradient in any point of Y: the proofs of these and the other basic properties stated throughout this paragraph, as well as a complete treatment of the underlying ε-subgradient theory, are beyond the scope of this work and will not be reported, the interested reader being referred to some of the available excellent books on Convex Analysis [Ro70] [HL93].

A particular class of concave functions is in fact the main target of the present work: *polyhedral functions* mostly arise from *Lagrangean Relaxations* of (Integer) Linear Programs, and have the form

$$\varphi(\boldsymbol{y}) = \min_{i=1..l} \{ b_i + \boldsymbol{y}\boldsymbol{h}_i \}$$

i.e. the pointwise minimum of a finite (but very large) set of linear functions. In this case, the black box typically returns the gradient $\boldsymbol{h}_i$ of one of the defining hyperplanes that are "tight" in $\bar{\boldsymbol{y}}$: this requires the solution of the possibly "difficult" optimization problem of choosing among the (exponentially many) facets that define the function.

For the sake of simplicity, the first two Chapters will focus on unconstrained NonDifferentiable Optimization problems, i.e. to the case $Y = \mathcal{R}^m$: by allowing $\varphi()$ to assume the value $-\infty$ somewhere,

this could be done without loss of generality, since any constrained problem can be turned into an unconstrained one by adding to $\varphi()$ the (concave) *characteristic function* of the set Y

$$\rho_Y(\boldsymbol{y}) = \begin{cases} 0 & \text{if } \boldsymbol{y} \in Y \\ -\infty & \text{otherwise} \end{cases}$$

The unconstrained case is interesting in itself, since for instance the Lagrangean Relaxation of equality constraints only gives rise to problems of this form: however, extensions to the constrained case will be discussed in §I.3.3, §I.3.5 and §I.4.3 for the case where Y is a polyhedron, that covers most practical applications.

The basic idea of Bundle methods is to collect all the information $\langle \varphi(\boldsymbol{y}_i), \boldsymbol{g}(\boldsymbol{y}_i) \rangle$ generated in the previous iterates in a set β, the *Bundle*, that is used at each step to compute a *tentative ascent direction* $\boldsymbol{d}$ along which the next points are generated. Since $\boldsymbol{d}$ is not guaranteed to be a direction of ascent, Bundle methods can usually decide whether to accept the step and move the *current point* $\bar{\boldsymbol{y}}$ (*SS*, a *Serious Step*), or rather use the new information to enlarge β and obtain a (hopefully) better $\boldsymbol{d}$ at the next iteration (*NS*, a *Null Step*): a very general Bundle scheme is stated in Figure 3.

Generic Bundle Algorithm

⟨ choose the initial $\bar{\boldsymbol{y}}$ and β ⟩;
do
⟨ find a tentative direction $\boldsymbol{d}$ ⟩;
⟨ move along $\boldsymbol{d}$, generating some new subgradients and a trial point $\boldsymbol{y} = \bar{\boldsymbol{y}} + \tau\boldsymbol{d}$ ⟩;
if(a large enough improvement has been obtained)
then $\bar{\boldsymbol{y}} = \boldsymbol{y}$; /* SS */
/* **else** it is a NS */
⟨ update β, adding new subgradients and deleting old ones ⟩;
⟨ update some algorithmic parameter, depending on the previous iterations ⟩;
while(⟨ some stopping condition ⟩);

Figure 3: a "generic" Bundle algorithm.

Such a general description allows the *Cutting Plane* algorithm to be included among the Bundle approaches: it is based on the simple observation that the *Cutting Plane model*

$$\bar{\varphi}_\beta(\boldsymbol{y}) = \min_{i \in \beta} \{ \varphi(\boldsymbol{y}_i) + \boldsymbol{g}(\boldsymbol{y}_i)(\boldsymbol{y} - \boldsymbol{y}_i) \}$$

(Figure 4) is a polyhedral upper approximation of $\varphi()$ - formed, when $\varphi()$ itself is polyhedral, of a subset of its facets - which is tight at least in every point $\boldsymbol{y}_i$: a straightforward idea is then to maximize the known $\bar{\varphi}_\beta()$ in place of the unknown $\varphi()$, and use (one of) the maximand(s) as the next trial point where to evaluate the function. An attractive feature of the CP algorithm is that a maximizer of $\bar{\varphi}_\beta()$ can be found by means of a Linear Programming problem: in fact, $\bar{\varphi}_\beta()$ can be rewritten with respect to $\boldsymbol{d} = \boldsymbol{y} - \bar{\boldsymbol{y}}$ (the displacement from the current point $\bar{\boldsymbol{y}}$), obtaining

$$\varphi_{\beta,\bar{\boldsymbol{y}}}(\boldsymbol{d}) = \min_{i \in \beta} \{ \alpha_i + \boldsymbol{d}\boldsymbol{g}_i \} + \varphi(\bar{\boldsymbol{y}})$$

where $\boldsymbol{g}_i = \boldsymbol{g}(\boldsymbol{y}_i)$, $\alpha_i = \varphi(\boldsymbol{y}_i) + \boldsymbol{g}(\boldsymbol{y}_i)(\bar{\boldsymbol{y}} - \boldsymbol{y}_i) - \varphi(\bar{\boldsymbol{y}}) \geq 0$ and $\varphi_{\beta,\bar{\boldsymbol{y}}}(\boldsymbol{d}) = \bar{\varphi}_\beta(\bar{\boldsymbol{y}} + \boldsymbol{d})$ (Figure 5);

then, the desired point is $\bar{\boldsymbol{y}} + \boldsymbol{d}^*$ where $\boldsymbol{d}^*$ is an optimal solution of

$(D_{\beta,\bar{y}})$ $\quad \max_{\boldsymbol{d}} \{ \varphi_{\beta,\bar{y}}(\boldsymbol{d}) \} = \max_{\boldsymbol{d},v} \{ v : v \le \boldsymbol{d}\boldsymbol{g}_i + \alpha_i \; i \in \beta \} + \varphi(\bar{\boldsymbol{y}})$.

Since $(D_{\beta,\bar{y}})$ has a limited number of constraints, it can be efficiently solved: note that, for the sake of simplicity, β will denote the set of pairs $\langle \alpha_i , \boldsymbol{g}(\boldsymbol{y}_i) \rangle$ as well as a set of their indices i, and $\varphi_\beta(\boldsymbol{d})$ will be used instead of $\varphi_{\beta,\bar{y}}(\boldsymbol{d})$ whenever the current point $\bar{\boldsymbol{y}}$ is clear from the context.

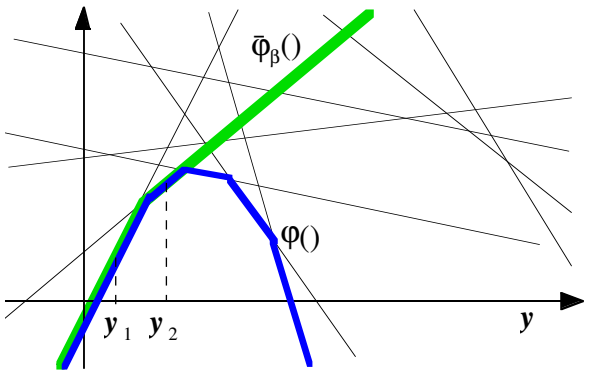


Figure 4: $\bar{\varphi}_\beta()$ approximates $\varphi()$, and is tight in $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$

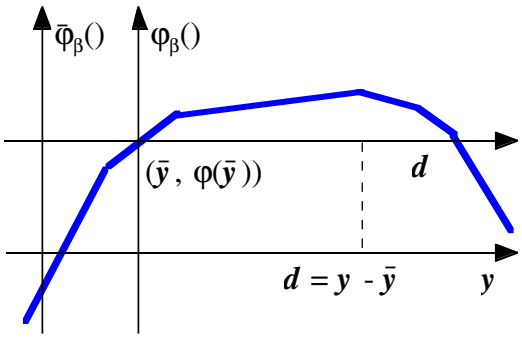


Figure 5: $\varphi_{\beta,\bar{y}}(\boldsymbol{d})$ is a translation of $\bar{\varphi}_\beta(\boldsymbol{y})$

If $\varphi_\beta(\boldsymbol{d}^*) = \varphi(\bar{\boldsymbol{y}} + \boldsymbol{d}^*)$ then $\boldsymbol{y} = \bar{\boldsymbol{y}} + \boldsymbol{d}^*$ is optimal for the entire problem (D), otherwise a new cutting plane $\langle \varphi(\boldsymbol{y}) , \boldsymbol{g}(\boldsymbol{y}) \rangle$ is generated and added to β to improve the CP model: this gives (in principle) a very simple globally convergent approach, requiring the solution of a (LP) at each step. Unfortunately, the CP approach is known to have a bad practical behaviour, mainly because of the "artificial" (usually box) constraints that have to be added to (D_β), at least in the initial iterations, to ensure that it has a finite solution: it is difficult to find "tight" such constraints that does not "cut away" $\boldsymbol{y}^*$, so that the new $\boldsymbol{y}$ is often unreasonably "far" from $\bar{\boldsymbol{y}}$, where $\varphi_\beta()$ does not correctly approximate $\varphi()$, resulting in a slow convergence.

A "dual" approach w.r.t. CP is based on the properties of the *subdifferential* $\partial\varphi(\bar{\boldsymbol{y}})$, i.e. the set of all the subgradients of $\varphi()$ in $\bar{\boldsymbol{y}}$: $\partial\varphi(\bar{\boldsymbol{y}})$ is a *closed* and *convex* set, and if $\varphi()$ is finite everywhere in a neighbourhood of $\bar{\boldsymbol{y}}$, then $\partial\varphi(\bar{\boldsymbol{y}})$ is also *compact*. The point-to-set mapping $\partial\varphi(\bar{\boldsymbol{y}})$ generalises the point-to-point mapping $\nabla\varphi(\bar{\boldsymbol{y}})$ since

- if $\varphi()$ is differentiable in $\bar{\boldsymbol{y}}$, then $\partial\varphi(\bar{\boldsymbol{y}}) = \{ \nabla\varphi(\bar{\boldsymbol{y}}) \}$;
- let $\varphi'(\bar{\boldsymbol{y}} , \boldsymbol{d})$ be the directional derivative at $\bar{\boldsymbol{y}}$ along the direction $\boldsymbol{d}$: the relation

  $$\varphi'(\bar{\boldsymbol{y}} , \boldsymbol{d}) = \nabla\varphi(\bar{\boldsymbol{y}})\boldsymbol{d}$$

  that holds if $\varphi()$ is differentiable in $\bar{\boldsymbol{y}}$, extends to

  $$\varphi'(\bar{\boldsymbol{y}} , \boldsymbol{d}) = \min_v \{ \boldsymbol{v}\boldsymbol{d} : \boldsymbol{v} \in \partial\varphi(\bar{\boldsymbol{y}}) \}$$

  so that $\varphi'(\bar{\boldsymbol{y}} , \boldsymbol{d})$ exists and it is *finite* if $\varphi()$ is finite everywhere in a neighbourhood of $\bar{\boldsymbol{y}}$ (in general, the concavity of $\varphi()$ implies that $\varphi'(\bar{\boldsymbol{y}} , \boldsymbol{d})$ exists for *any* $\bar{\boldsymbol{y}}$ and $\boldsymbol{d}$, but it can be $-\infty$);
- $\nabla\varphi(\bar{\boldsymbol{y}})$ (when existent) is the *steepest ascent direction* for $\varphi()$ in $\bar{\boldsymbol{y}}$, i.e.

  $$\nabla\varphi(\bar{\boldsymbol{y}}) = \operatorname{argmax}_{\boldsymbol{d}} \{ \varphi'(\bar{\boldsymbol{y}} , \boldsymbol{d}) : \| \boldsymbol{d} \| = 1 \}$$

but more in general

$(Q_{\bar{y}})$ $\quad \operatorname{argmax}_d \{ \varphi'(\bar{y}, d) : \|d\| = 1 \} = \operatorname{argmin}\{ \|g\| : g \in \partial\varphi(\bar{y}) \};$

– finally, from the above relations it follows that

$$y^* \text{ is } \textit{optimal} \text{ for } \varphi() \Leftrightarrow \mathbf{0} \in \partial\varphi(y^*) \Leftrightarrow 0 = \min\{ \|g\| : g \in \partial\varphi(y^*) \}.$$

At a first sight, nondifferentiable concave functions seem not to be much harder to maximize than differentiable ones: first-order information about $\varphi()$ of (apparently) the same "strength" as $\nabla\varphi()$ can be obtained at the "only" cost of solving the Least-Square *steepest ascent direction finding* subproblem $(Q_{\bar{y}})$ on the convex set $\partial\varphi()$. However, the rationale under the "black box" form of $\varphi()$ is exactly that it is impossible, or too costly, to work with the whole $\partial\varphi()$: the simplest idea is then to use *any* subgradient, i.e. the one returned by the black box, as an approximation of an optimal solution of $(Q_{\bar{y}})$. The result is the classic *Subgradient Algorithm* [Po77] [BNS87], that directly generalises the Steepest Descent method for the smooth case: its main advantages are its low cost and straightforward implementations, that have made it very popular among researchers in practical optimization. In the Bundle terminology, β here is made of the last subgradient only, and therefore movements to the trial point y must always be performed in order to avoid cycling when d is not a direction of ascent; the stepsize along the current direction is usually taken according to a predefined rule, several of which have been proposed in the literature [HVC74] [CFM75] [Br95], rather than dynamically adjusted via a Line Search. Unfortunately, the rules that work better in practice (e.g. geometric decrease of the stepsize every p iterations) does not usually guarantee convergence to the optimal solution; furthermore, although the method can be extended by taking into account the previous direction [Cr76] [Br91] or the last two subgradients [KU93], its fundamental Markovian nature heavily limits its efficiency: at the k-th step, almost no trace is left of what happened at steps $0 \,..\, k-1$.

The main theoretical problem of Subgradient algorithms is that (see Figure 2) subgradients are not guaranteed to be directions of ascent: in principle, however, this can (in theory) be easily detected by an exact Line Search, as if the current direction is not of ascent, the LS will just return the current point $\bar{y}$. During such a LS, a new subgradient at $\bar{y}$ (different from the current one) could be generated: this suggest to resort to some form of *inner approximation* of the subdifferential, where the Bundle β is meant to contain some subgradients of the current point. A *tentative* steepest ascent direction could be found by solving $(Q_{\bar{y}})$ restricted to the convex hull of the known subgradients in $\bar{y}$, i.e.

$$(\Delta_\beta) \qquad \min_\theta \left\{ \left\| \sum_{j \in \beta} g_j \theta_j \right\| : \theta \in \Theta \right\} = \min_g \{ \|g\| : g \in \operatorname{conv}(\{ g_i : i \in \beta \}) \}$$

where $\Theta = \{ \theta : \sum_{j \in \beta} \theta_j = 1 , \theta \geq \mathbf{0} \}$ is the unitary simplex in the $|\beta|$-dimensional space: if $d = \sum_{j \in \beta} g_j \theta_j$ is not of ascent, the new subgradient is used to refine the current approximation of $\partial\varphi(\bar{y})$. Such an approach is only *conceptual*, however, since an exact LS may not finitely terminate on generic functions - although it would on polyhedral ones - and, even if it did, there is no guarantee that a new subgradient at $\bar{y}$ would be generated: note that calling again the black box on $\bar{y}$ does not solve the problem. However, one may hope that, if the LS terminates "close" to $\bar{y}$, then the last subgradient will be "close" to a subgradient at $\bar{y}$: this idea leads to the *Conjugated Subgradient* method [Wo75], that uses (Δ_β) a specialized inexact Line Search along d to find either a significantly better

point $\boldsymbol{y}$, or a new subgradient $\boldsymbol{g}$ which is *not* guaranteed to be a subgradient in $\bar{\boldsymbol{y}}$, but to yield a different search direction $\boldsymbol{d}$ at the next iteration; since it is obtained "close" to $\bar{\boldsymbol{y}}$, it still gives relevant information on the behaviour of $\varphi()$ around $\bar{\boldsymbol{y}}$. A drawback of such an approach is that every time a good improvement is obtained and the current point is moved, β should be emptied, loosing all the information about $\varphi()$: since the elements of β do not really need to be subgradients of the new current point, a more attractive alternative is to keep them, provided that some sort of *reset rule* avoids outdated information to interfere with the determination of a correct $\boldsymbol{d}$. Several such rules may be imagined, such as the *exponential smoothing* technique where at each step an old $\boldsymbol{g}_i$ is multiplied by a factor $\mu^{A(i)}$, where $\mu < 1$ and $A(i)$ is its *age* (how many iterations ago it was found): however, a more robust approach calls for theoretically justifiable rules, that can be obtained by appropriately defining *approximate subgradients*.

An ε-*subgradient* (of $\varphi()$ in $\bar{\boldsymbol{y}}$) has to satisfy the obvious inequality

$$\varphi(\boldsymbol{y}) \le \varphi(\bar{\boldsymbol{y}}) + \boldsymbol{g}(\boldsymbol{y} - \bar{\boldsymbol{y}}) + \varepsilon \qquad \forall\ \boldsymbol{y}$$

i.e. to be the gradient of a supporting hyperplane $\boldsymbol{g}$ of the epigraph of $\varphi()$ in $(\bar{\boldsymbol{y}},\ \varphi(\bar{\boldsymbol{y}}) + \varepsilon)$ (Figure 6): the previous results for the special case $\varepsilon = 0$ can be generalized by using the notions of ε-*subdifferential* set $\partial_\varepsilon\varphi(\bar{\boldsymbol{y}})$ and ε-*ascent direction* for $\varphi()$ in $\bar{\boldsymbol{y}}$, that is a direction $\boldsymbol{v}$ such that

$$\exists\ t > 0 \text{ s.t. } \varphi(\bar{\boldsymbol{y}} + t\boldsymbol{v}) - \varphi(\bar{\boldsymbol{y}}) > \varepsilon$$

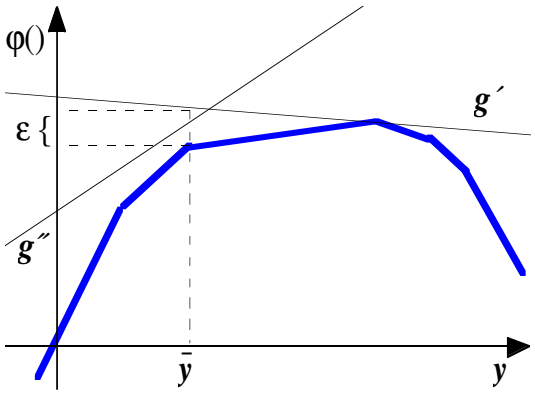


Figure 6: two ε-subgradients for $\varphi()$ in $\bar{\boldsymbol{y}}$: $\boldsymbol{g}'$ gives the (right) ε-directional derivative in $\bar{\boldsymbol{y}}$

i.e. an improvement strictly larger than ε is *possible* along $\boldsymbol{v}$. A strictly related concept is that of ε-*directional derivative* of $\varphi()$ in $\bar{\boldsymbol{y}}$ along $\boldsymbol{d}$, that is, roughly speaking, the slope of the steepest line contained in the restriction of $\varphi()$ along $\boldsymbol{d}$, passing from $(\bar{\boldsymbol{y}}\ ,\ \varphi(\bar{\boldsymbol{y}}) + \varepsilon)$ and touching the epigraph of $\varphi()$ somewhere (Figure 6): more formally

$$\varphi'_\varepsilon(\bar{\boldsymbol{y}}\ ,\boldsymbol{d}) = \sup_{t\,>\,0} \frac{\varphi(\bar{\boldsymbol{y}} + t\boldsymbol{v}) - \varphi(\bar{\boldsymbol{y}}) - \varepsilon}{t}$$

(the formula justifies for the name), and it can be easily seen that $\boldsymbol{d}$ *is of* ε-*ascent* $\Leftrightarrow \varphi'_\varepsilon(\bar{\boldsymbol{y}}\ ,\boldsymbol{d}) > 0$.

The previous results gracefully extend to the new case, i.e.

- $\varphi'_\varepsilon(\bar{\boldsymbol{y}}\ ,\boldsymbol{d}) = \min_{\boldsymbol{v}}\{\ \boldsymbol{v}\boldsymbol{d} : \boldsymbol{v} \in \partial_\varepsilon\varphi(\bar{\boldsymbol{y}})\ \}$;
- $(Q^\varepsilon_{\bar{\boldsymbol{y}}})$ $\operatorname{argmin}_{\boldsymbol{g}}\{\ \|\boldsymbol{g}\| : \boldsymbol{g} \in \partial_\varepsilon\varphi(\bar{\boldsymbol{y}})\ \} = \operatorname{argmax}_{\boldsymbol{d}}\{\ \varphi'_\varepsilon(\bar{\boldsymbol{y}}\ ,\boldsymbol{d}) : \|\boldsymbol{d}\| = 1\ \}$;
- $\bar{\boldsymbol{y}}$ is ε-optimal for $\varphi() \Leftrightarrow \boldsymbol{0} \in \partial_\varepsilon\varphi(\bar{\boldsymbol{y}}) \Leftrightarrow 0 = \min_{\boldsymbol{g}}\{\ \|\boldsymbol{g}\| : \boldsymbol{g} \in \partial_\varepsilon\varphi(\bar{\boldsymbol{y}})\ \}$;

but ε-subgradients also enjoy the so-called *information transport property*

$$\boldsymbol{g} \in \partial_\varepsilon\varphi(\bar{\boldsymbol{y}}) \ \Rightarrow\ \boldsymbol{g} \in \partial_\mu\varphi(\bar{\boldsymbol{y}}) \quad \forall\ \boldsymbol{y}\ \forall\ \mu \ge \alpha = \varphi(\bar{\boldsymbol{y}}) + \boldsymbol{g}(\boldsymbol{y} - \bar{\boldsymbol{y}}) - \varphi(\boldsymbol{y}) + \varepsilon$$

saying, in plain words, that a vector $\boldsymbol{g}$ belonging to the ε-subdifferential of $\varphi()$ in some point $\bar{\boldsymbol{y}}$ also belongs to the α-subdifferential of $\varphi()$ in any other point $\boldsymbol{y}$, where $\alpha \ge 0$ (known as *linearization error*) can be calculated from $\boldsymbol{g}$ and $\varphi(\boldsymbol{y})$. Hence, the first-order information at hand about $\varphi()$ in $\bar{\boldsymbol{y}}$

can be "translated" into an information about a completely unrelated point $\boldsymbol{y}$, at the cost of "deteriorating" it of a known measure α. Therefore, one possibility is just to fix some $\varepsilon > 0$ and solve (Δ_β) restricted to the (known) ε-subgradients of the current point

$$\min \{ \| \boldsymbol{g} \| : \boldsymbol{g} \in \text{conv}(\{ \boldsymbol{g}_i : \alpha_i \leq \varepsilon , i \in \beta \}) \}$$

i.e. the steepest ε-ascent direction finding problem restricted to an inner linearization of the ε-subdifferential $\partial_\varepsilon \varphi(\bar{\boldsymbol{y}})$: the very first algorithms that were referred to as *Bundle methods* [Le75] [Le77] [BNS87] [HL93 Algorithm XIV.3.4.2] can be seen as a "smoothing" of the above reset rule, since they are based on the same ideas but on the slightly different (QP)

$$(\Delta_\beta^\varepsilon) \quad \min_\theta \{ \| \Sigma_{j \in \beta} \boldsymbol{g}_j \theta_j \| : \Sigma_{j \in \beta} \alpha_j \theta_j \leq \varepsilon , \theta \in \Theta \}$$

where, rather than eliminating all the $\boldsymbol{g}_i$ with $\alpha_i > \varepsilon$, the α_i are used as "weights" of the corresponding vector, and the constraint ensures that any $\boldsymbol{g}_i$ with a "large" α_i will have a "small influence" (multiplier θ_i) on the direction. Other than being heuristically clear, this approach has a neat theoretical justification [HL93 Example 3.5.3] since the ε-subdifferential of the polyhedral function

$$\varphi(\boldsymbol{y}) = \min_{i = 1 \,..\, l} \{ b_i + \boldsymbol{y} \boldsymbol{h}_i \}$$

is the polytope

$$\partial_\varepsilon \varphi(\bar{\boldsymbol{y}}) = \{ \boldsymbol{g} = \Sigma_{i = 1..l} \boldsymbol{h}_i \theta_i : \Sigma_{i = 1..l} \xi_i \theta_i \leq \varepsilon \}$$

where $\xi_i = \bar{\boldsymbol{y}} \boldsymbol{h}_i + b_i - \varphi(\bar{\boldsymbol{y}})$; therefore, if each $\boldsymbol{h}_i$ is read as the subgradient of $\varphi()$ in a point $\boldsymbol{y}_i$ such that $\varphi(\boldsymbol{y}_i) = b_i + \boldsymbol{y}_i \boldsymbol{h}_i$, ξ_i is exactly the linearization error α_i w.r.t. the current point $\bar{\boldsymbol{y}}$, i.e. $(\Delta_\beta^\varepsilon)$ corresponds to the *steepest ε-ascent direction finding problem* (at $\bar{\boldsymbol{y}}$) if the CP-model $\bar{\varphi}_\beta()$ is used in place of the real function $\varphi()$. The critical part of this approach is the introduction of the algorithmic parameter ε, that must be dynamically updated in order to achieve good practical performances: unfortunately, it is not clear what sort of rules should be devised for the task. Therefore, the more recent approaches [Ki90] [SZ92] [HL93 Algorithm XV.3.3.4] use the further "smoothing"

$$(\Delta_{\beta t}) \quad \min_\theta \{ 1/2 \, t \| \Sigma_{j \in \beta} \boldsymbol{g}_j \theta_j \|^2 + \Sigma_{j \in \beta} \alpha_j \theta_j : \theta \in \Theta \}$$

that may viewed as a Lagrangean relaxation of $(\Delta_\beta^\varepsilon)$ where a (fixed) Lagrangean multiplier $1/t$ is used to dualize the constraint $\Sigma_{j \in \beta} \alpha_j \theta_j \leq \varepsilon$; actually, such a relaxation would be

$$\min_\theta \{ 1/2 \| \Sigma_{j \in \beta} \boldsymbol{g}_j \theta_j \|^2 + (1/t) \Sigma_{j \in \beta} \alpha_j \theta_j : \theta \in \Theta \} \; [- \varepsilon t]$$

that differs from $(\Delta_{\beta t})$ for a multiplicative factor of t in the objective function, but the Khun-Tucker conditions easily show that the two problems are essentially the same, i.e. they have the same optimal primal solution θ and the respective dual optimal solutions $(v, \boldsymbol{d})$ are scaled of a factor of t. Historically, $(\Delta_\beta^\varepsilon)$ was considered the "basic" (QP) subproblem to be solved, and in fact it is possible to develop the whole theory of Bundle algorithms in terms of $(\Delta_\beta^\varepsilon)$ rather than of $(\Delta_{\beta t})$ [HL93]: this choice had a practical rationale, since the constrained least square problem $(\Delta_\beta^\varepsilon)$ was considered easier to solve than the general (QP) with one constraint $(\Delta_{\beta t})$ [Ki85a Introduction to Chapter 7] - in fact, ten years separate [Mi77] from [Ki86] where the first specialized algorithms for the two different problems have been proposed. As shown in Chapter I.3, this is not true any longer, and besides (Cf. §I.1.2) $(\Delta_\beta^\varepsilon)$ can be solved for any fixed ε by means of $(\Delta_{\beta t})$: hence, the development in terms of $(\Delta_{\beta t})$

seem to be preferable for several reasons, the main being that rules for choosing t are apparently easier to develop than rules for choosing ε. Such rules (the *t-strategies*), which are the critical part of any Bundle algorithm, will be extensively discussed in Chapter I.5: however, all of them are related to the *trust radius nature* of the parameter t, that becomes apparent by writing the Quadratic Dual of $(\Delta_{\beta t})$

$$(\Pi_{\beta t}) \qquad \max_{d,v} \{ v - 1/2t \| \boldsymbol{d} \|^2 : v \le \boldsymbol{d}\boldsymbol{g}_j + \alpha_j \; j \in \beta \}.$$

$(\Pi_{\beta t})$ is equivalent to $(\Delta_{\beta t})$ in the sense that, being θ the optimal solution of $(\Delta_{\beta t})$, the optimal solutions $\boldsymbol{d}$ and v of $(\Pi_{\beta t})$ can be computed as

$$\boldsymbol{d} = t\boldsymbol{z} \qquad \text{where} \qquad \boldsymbol{z} = \Sigma_{j \in \beta} \boldsymbol{g}_j \theta_j$$

$$v = t \| \boldsymbol{z} \|^2 + \sigma \qquad \text{where} \qquad \sigma = \Sigma_{j \in \beta} \alpha_j \theta_j$$

i.e. the new tentative point $\boldsymbol{y}$ is found along the *direction* $\boldsymbol{z}$ (a convex combination of the currently available subgradients), using t as *predefinite stepsize*: this naturally leads to the idea of *Bundle/Trust Region* approaches, where t is increased or decreased according to the outcome of the last (few) step(s). In case of a SS, increasing t may be preferred to moving to $\boldsymbol{y}$ whenever there is the chance that $\varphi_\beta()$ be a "good" approximation of $\varphi()$ also outside the current "trust region"; in case of a NS, decreasing t may be preferred to inserting the new $\boldsymbol{g}$ if the latter is not believed to carry "reliable information" about $\varphi()$ in a neighbourhood of $\bar{\boldsymbol{y}}$ (e.g. due to a large α). However, Bundle methods of this form are *not* pure Trust Region algorithms for several reasons, the simplest being that t is not a trust radius since $\boldsymbol{z}$ is not in principle of unitary norm, and $\| \boldsymbol{z} \|$ depends on t in a "complicated" way so that the *actual displacement* $t \| \boldsymbol{z} \|$ is only known *a posteriori*: furthermore, using a Line Search within such an approach is still possible, and may also be advantageous in some cases.

Interestingly enough the Conjugated Subgradient and the Cutting Plane algorithms can be obtained as the "extreme cases" of a Bundle method using $(\Delta_{\beta t})$: in fact, CP and CS are respectively the "limit for $t \to 0$" and the "limit for $t \to \infty$" of the above scheme, since for example the - possibly unbounded - optimal solution of (D_β) is just the limit for $t \to 0$ of the optimal solution of $(\Pi_{\beta t})$. A good way of seing this equivalence is rewriting $(\Pi_{\beta t})$ as

$$(\Pi_{\beta t}) \qquad \max_d \{ \varphi_\beta(\boldsymbol{d}) - 1/2t \| \boldsymbol{d} \|^2 \}$$

i.e. the same as (D_β) but for a simple quadratic *stabilizing term*: in fact, Bundle algorithms are often regarded to as "stabilized" versions of the CP algorithm that avoid its inherent "instability" problems, i.e. the fact that $\bar{\boldsymbol{y}} + \boldsymbol{d}^*$ can be very "far" from $\bar{\boldsymbol{y}}$, even if the latter is very near to the the optimum. The stabilizing term introduces a measure of distance from the *prox. center* $\bar{\boldsymbol{y}}$, discouraging directions leading to "far away" points: assuming that $\varphi_\beta()$ has been constructed as to be a "local" model of $\varphi()$, this can be interpreted as avoiding "far away" points where $\varphi_\beta()$ is probably a "bad" approximation of $\varphi()$.

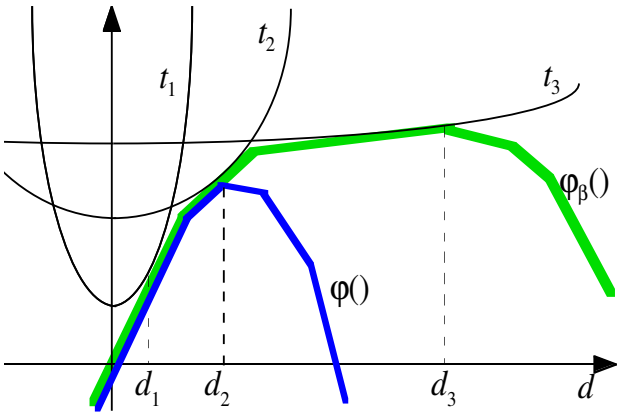


Figure 7: effect of the stabilizing term; for $t_1 < t_2 < t_3$, d_i are the optimal solutions

Again, t is a parameter that measures the "trust" in $\varphi_\beta()$ as a model of $\varphi()$ far from the current point $\bar{\boldsymbol{y}}$: the direction $\boldsymbol{d}$ is a "function" of t and, for sufficiently large values of t, $(\Pi_{\beta l})$ is equivalent to the "naive" (D_β) (Figure 7), provided that the latter problem has finite maximum.

Hence, Bundle methods belong to the class of *Proximal Point* approaches, originally introduced in [Ro76], that in the "modern" view are connected to the *Moreau-Yosida Regularization* of $\varphi()$

$$\phi_{\boldsymbol{M},t}(\boldsymbol{y}) = \max_{\boldsymbol{d}} \{ \varphi(\boldsymbol{y}+\boldsymbol{d}) - 1/2t \| \boldsymbol{d} \|_{\boldsymbol{M}}^2 \}$$

where $\boldsymbol{M}$ is a positive semi-definite matrix intended to take the place of $\nabla^2\varphi()$ (whatever the Hessian of a nonsmooth function could mean); $\phi_{\boldsymbol{M},t}()$ is in some sense the "ideal" version of $\varphi()$, since it has the same set of minimizers but it is differentiable, so that a conceptually attractive way of maximizing $\varphi()$ would be that of applying any (efficient) smooth optimization algorithm to $\phi_{\boldsymbol{M},t}()$. Unfortunately, the computation of just a single value of $\phi_{\boldsymbol{M},t}(\boldsymbol{y})$ requires the solution of a NonDifferentiable Optimization problem, in terms of the original "black box", in principle as difficult as the overall maximization of $\varphi()$: however, it is quite easy to see (Cf. §I.2.2) that a sequence of consecutive Null Steps is just a Cutting Plane approach applied to the computation of $\phi_{\boldsymbol{M},t}(\bar{\boldsymbol{y}})$, so that the "bundling" mechanism provides a natural framework for algorithms based on these concepts.
In other words, any Bundle algorithm can be seen as a "two-level" approach to the maximization of $\phi_{\boldsymbol{M},t}()$, where sequences of consecutive NS aim at approximately compute $\phi_{\boldsymbol{M},t}(\bar{\boldsymbol{y}})$ while Serious Steps use the resulting information for the actual maximization: hence, the challenge is to adapt the efficient Quasi-Newton methods, that could be directly used if $\phi_{\boldsymbol{M},t}()$ were known exactly, to this new setting, where the smallest possible number of $\varphi()$ computations (NS) shall be used between two "upper level" moves (SS). Quite a lot of work is currently being done on this subject, as attested by several recent papers (e.g. [Mi82] [Mi91] [GM92] [BGL95] [FQ96] [LS96a] [LS96b] [LS96c] [Mi96] and many others, few of which already in the open literature): part of this renewed interest is due to some results that show how superlinearly convergent Quasi-Newton-type methods can be obtained in spite of the fact that $\phi_{\boldsymbol{M},t}()$ is *not twice continuously differentiable*. In fact, under some assumptions on the "original" function $\varphi()$, the directional derivatives of $\phi_{\boldsymbol{M},t}()$ can be shown to be *semismooth*, and this property can be enough for superlinear convergence to hold if the matrix $\boldsymbol{M}$ is properly handled; although some encouraging computational results have already been presented, the state of the art on this subject is very rapidly evolving, and a discussion of the related technical details will not be issued here.

Different forms of Proximal Point method has been recently proposed, where the quadratic proximal term is replaced by some different function, entropy-like [IST94] or even nonsmooth such as the L_1 or L_∞ norms [KCL95]: the latter is related to the new family of Bundle methods proposed in Chapter I.4, and will be discussed in some more details in §I.4.1.

Another recent entry in the family of Bundle methods is the *Proximal Level* method [HL93 Algorithm 2.3.1] [BKL95] [Ki95] [LNN95], that uses

$$(\Lambda_{\beta l}) \qquad \min \{ \| \boldsymbol{d} \|^2 : \varphi_\beta(\boldsymbol{d}) \geq l \}$$

as the direction finding problem, where l is the algorithmic parameter that plays a role similar to that of t: Proximal Level methods have been reported to be less sensitive to the choice of the l parameter than other forms of Bundle method be with theirs, but no decisive evidence of clear superiority seem to

exist yet.

Finally, a quite different form of Bundle approach uses the concept of *analytic centre* to locate the next trial point [GHV92] [RV92] [GHV93] [AV95] [Ne95] [AK96] [MGV96]: applied to the optimization of a concave function, the methods look at the polyhedron

$$P_{\bar{y},\beta} = \{ (v, \boldsymbol{y}) : \varphi_\beta(\boldsymbol{d}) \geq v \geq \varphi(\bar{y}) \} \subseteq \mathcal{R}^{m+1}$$

and compute its analytic centre, i.e. the point (π, ω) maximizing the product of the slacks from the boundaries, or equivalently the solution of

$$(A_{\bar{y},\beta}) \qquad \max_{d,v} \{ \Sigma_{j \in \beta} \ln(\boldsymbol{d}\boldsymbol{g}_j + \alpha_j - v) + \ln(v) : (v, \boldsymbol{d}) \in \mathcal{R}^{m+1} \}.$$

The analytic centre (π, ω) lies in the interior of $P_{\bar{y},\beta}$, and it is related in a well-known way to Interior Point methods for linearly constrained convex optimization [CLM93]: the ω component is then given to the "black box" to compute $\langle \varphi(\omega), \boldsymbol{g}(\omega) \rangle$, that can be interpreted as a separating hyperplane (a *cut*) in $\mathcal{R}^{m+1}$. Hence, these methods essentially keep "shrinking" $P_{\bar{y},\beta}$ by adding new cuts, and are in fact somehow better understood as algorithms for proving nonemptyness of a convex body given by a separation oracle: the properties of the analytic centre, together with a judicious choice of the cuts to be conserved, allow to derive polynomial bounds on the number of iterations required to reduce the volume (or the "width") of $P_{\bar{y},\beta}$ down to any required threshold, i.e. to achieve any required precision. Clearly, the solution of the $(A_{\bar{y},\beta})$ subproblem can be in principle a heavy task, but it is possible to exploit the analytic centre of the previous iterate to "warm-start" a Newton method, that can be shown to converge in a constant (and very small) number of iterations to a "close enough" approximation of the new analytic centre: the resulting overall complexity estimates are quite attractive, and the methods have an interesting behaviour in practice [GGS94], especially if "hybridized" with traditional CP techniques [MGV96] (Cf. Sec. II.1.4.2).

The remainder of this Chapter will essentially focus on "standard" Bundle algorithms, i.e. on methods that use subproblems (similar to) $(\Pi_{\beta t})$ and $(\Delta_{\beta t})$: these (QP)s have emerged from the very beginning [Le77] [Le89] [Le91] as interesting candidates for the direction finding subproblem within Bundle methods, probably due to the fact that they can be viewed from a surprisingly high number of different viewpoints, each one possibly suggesting new ways of approaching some of the open questions in the development of a good overall algorithm. In fact, a part of this work is exactly aimed at describing the interpretations of $(\Pi_{\beta t})$ and $(\Delta_{\beta t})$ when $\varphi()$ is a polyhedral function: taking into account all these viewpoints naturally leads to the definition of a new, broader class of Bundle methods and also to the development of new t management strategies.

However, the "dual" viewpoint of Bundle methods is very important for the understanding of the algorithms and for proving results: for instance, it is easy to show that, for any $\theta \in \Theta$, the vector z belongs to the σ-subdifferential of the current point $\bar{y}$, and therefore the relation

$$\| z \| = 0 \qquad \Rightarrow \qquad \bar{y} \text{ is } \sigma\text{-optimal}$$

can be easily understood and proved in terms of the above ε-subgradients theory. It also gives a rationale for the "standard" stopping conditions used within Bundle algorithms, i.e.

$$\| z \|^2 \leq \varepsilon_Q \qquad \text{and} \qquad \sigma \leq \varepsilon_L$$

where ε_Q and ε_L are two fixed tolerances: however, choosing a suitable value for ε_Q can be a delicate task, and a full understanding of the above stopping conditions requires some analysis. Let $\boldsymbol{y}^*$ be an optimal solution of (D): the "ideal" stopping criterion is

$$\varphi(\boldsymbol{y}^*) - \varphi(\bar{\boldsymbol{y}}) \le \varepsilon$$

i.e. ε-optimality for a fixed absolute tolerance ε, but this is not an implementable rule since in general the value of $\varphi(\boldsymbol{y}^*)$ is unknown. Since $z \in \partial_\sigma \varphi(\bar{\boldsymbol{y}})$, i.e. $\varphi(\boldsymbol{y}^*) \le \varphi(\bar{\boldsymbol{y}}) + (\boldsymbol{y}^* - \bar{\boldsymbol{y}})z + \sigma$,

$$\sigma + (\boldsymbol{y}^* - \bar{\boldsymbol{y}})z \le \varepsilon$$

would an even stronger stopping rule, if only $\boldsymbol{y}^*$ was known; analogously, by defining

$$t^*(\boldsymbol{y}, z) = \| \boldsymbol{y}^* - \boldsymbol{y} \| / \| z \|$$

(the distance from $\boldsymbol{y}$ and the optimum, taking $\| z \|$ as the unitary distance), the even stronger

$$\sigma + t^*(\bar{\boldsymbol{y}}, z) \| z \|^2 \le \varepsilon$$

could be turned into the "standard" form by letting $\varepsilon_L = \varepsilon / 2$ and choosing at each step some $\varepsilon_Q \le \varepsilon_L / t^*(\bar{\boldsymbol{y}}, z)$, if the function $t^*()$ would be known. Since they all contains an unknown quantity, these formulae only give an idea on how ε_L and ε_Q should be set, saying that ε_Q should be related to both ε_L and to some measure of the distance to the optimum measured "along" z: however, they suggest an alternative approach, i.e. to stop when

$$\sigma + t^* \| z \|^2 \le \varepsilon$$

where t^* is a user-provided fixed parameter, intended to serve as an upper approximation of the function $t^*()$. Usually, a viable such t^* can be easily choosen by observing a few runs of the algorithm, and seems to be fairly stable for problems within the same class: furthermore, this stopping criteria will show to be useful in Chapter I.2 when dealing with convergence. In practice, the *absolute* tolerance ε is better replaced by a *relative* tolerance, e.g. by using $\varepsilon | \varphi(\bar{\boldsymbol{y}}) |$ as the right hand side of the inequality, where $\varphi(\bar{\boldsymbol{y}})$ serves as an (usually pretty good) estimate of the magnitude of $\varphi(\boldsymbol{y}^*)$; the only event to take care of is $\varphi(\boldsymbol{y}^*) \approx \varphi(\bar{\boldsymbol{y}}) \approx 0$, that can be faced for instance by using $\max\{ | \varphi(\bar{\boldsymbol{y}}) |, 1 \}$ instead of $| \varphi(\bar{\boldsymbol{y}}) |$.

I.1.2 The Primal viewpoint

Part of the interest of NDO algorithms comes from the fact that they can be used as a tool for efficiently solving problems of an apparently very different nature: the typical example is

(H) $$\min\{ \boldsymbol{c}\boldsymbol{x} : \boldsymbol{x} \in X, \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \}$$

where $X \subseteq \mathcal{R}^n$ is a "simple" compact set and $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ are "complicating" constraints, in the sense that

(P_μ) $$\min\{ \mu\boldsymbol{x} : \boldsymbol{x} \in X \}$$

can be "efficiently" solved for any choice of μ while the whole (H) is significantly harder. In many approaches, (H) is "replaced" by the convex program

(P) $$\min\{ \boldsymbol{c}\boldsymbol{x} : \boldsymbol{x} \in \overline{X}, \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \}$$

where $\bar{X} = \text{conv}(\boldsymbol{x}_1, .., \boldsymbol{x}_p)$, each $\boldsymbol{x}_i$ being an extreme point of X; this general framework covers several different cases, such as

- X is a structured polyhedron itself, such as a flow and/or a separable polytope, so that (H) = (P) is a structured (LP) with "complicating" constraints;
- X is a discrete set, (P_μ) is a relaxation of (H) that can be efficiently solved (either to optimality or with an heuristic) and (P) is a relaxation of (H) that is used to derive lower bounds;
- X is a generic convex set that can be finitely inner approximated for the purpose of (approximately) solving (H);
- combinations of the above.

Following the classic Dantzig-Wolfe decomposition approach [DW60], the Master Problem is

$$(M) \qquad \min_\theta \{ \boldsymbol{c}(\Sigma_{j=1..p} \boldsymbol{x}_j\theta_j) : \boldsymbol{A}(\Sigma_{j=1..p} \boldsymbol{x}_j\theta_j) = \boldsymbol{b}, \theta \in \Theta \}$$

where Θ is the unitary simplex of proper dimension: (M) is obviously equivalent to (P), where each $\boldsymbol{x} \in \bar{X}$ is explicitly described as a convex combination of the *vertices* $\boldsymbol{x}_j$ using (non unique) *combinators* θ. The fundamental strategy of the DW algorithm is to solve (M) by generating only a(n hopefully small) subset of the vertices, so that from now on the subscript β will be meant to restrict something to the vertices in the subset β, as in

$$(M_\beta) \qquad \min_\theta \{ \boldsymbol{c}(\Sigma_{j\in\beta} \boldsymbol{x}_j\theta_j) : \boldsymbol{A}(\Sigma_{j\in\beta} \boldsymbol{x}_j\theta_j) = \boldsymbol{b}, \theta \in \Theta \} = \min_\theta \{ \boldsymbol{cx} : \boldsymbol{x} \in \bar{X}_\beta, \boldsymbol{Ax} = \boldsymbol{b} \}.$$

The linear dual problem of (M_β) is

$$(L_\beta) \qquad \max_{\boldsymbol{y},v} \{ \boldsymbol{yb} + v : v \le (\boldsymbol{c} - \boldsymbol{yA})\boldsymbol{x}_i \;\; i \in \beta \}$$

and its optimal solutions (v, $\boldsymbol{y}$) are used in the DW approach for the so called "mechanised pricing", i.e. for obtaining an optimal solution $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{y})$ to the *pricing subproblem*

$$(P_{\boldsymbol{c}-\boldsymbol{yA}}) \qquad \min_{\boldsymbol{x}} \{ (\boldsymbol{c} - \boldsymbol{yA})\boldsymbol{x} : \boldsymbol{x} \in X \} \; [+ \boldsymbol{yb}]$$

that gives a new "tentative optimal" vertex of $\bar{X}$: if the corresponding constraint $v \le (\boldsymbol{c} - \boldsymbol{yA})\boldsymbol{x}$ is satisfied, then (v, $\boldsymbol{y}$) are clearly dual optimal solutions for the whole (L) and $\bar{\boldsymbol{x}} = \Sigma_{j\in\beta} \boldsymbol{x}_j\theta_j$ (θ being the optimal solution of (M_β)) is optimal for (P), otherwise $\boldsymbol{x}$ is added to β and the process is repeated. It is well-known that the DW approach is exactly equivalent to the CP approach applied to the Lagrangean Dual of (P) with respect to the complicating constraints $\boldsymbol{Ax} = \boldsymbol{b}$: in fact, by letting $\varphi(\boldsymbol{y}) = z(P_{\boldsymbol{c}-\boldsymbol{yA}})$, (L) can be written as the unconstrained maximization of the concave polyhedral function $\varphi()$ and $\boldsymbol{g}(\bar{\boldsymbol{y}}) = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}(\bar{\boldsymbol{y}})$ is a subgradient for $\varphi()$ in $\bar{\boldsymbol{y}}$. Hence, the optimal dual prices $\boldsymbol{y}$ are used from a *primal viewpoint* to obtain a new $\boldsymbol{x}_i = \boldsymbol{x}(\boldsymbol{y}_i)$ to enrich the inner approximation $\bar{X}_\beta$, and from the *dual viewpoint* to obtain a new $\boldsymbol{g}_i = \boldsymbol{g}(\boldsymbol{y}_i)$ to enrich the CP-model $\varphi_\beta()$. For any dual vector $\boldsymbol{y}$, $\varphi(\boldsymbol{y})$ can be written as $\varphi(\boldsymbol{y}) = \boldsymbol{cx}(\boldsymbol{y}) + \boldsymbol{yg}(\boldsymbol{y})$, and the linearization errors α_j of $\varphi()$ w.r.t. any current point $\bar{\boldsymbol{y}}$ are

$$\alpha_j = \varphi(\boldsymbol{y}_j) + (\bar{\boldsymbol{y}} - \boldsymbol{y}_j)\boldsymbol{g}_j - \varphi(\bar{\boldsymbol{y}}) = (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})(\boldsymbol{x}_j - \boldsymbol{x}(\bar{\boldsymbol{y}}))$$
$$= (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\boldsymbol{x}_j - \varphi(\bar{\boldsymbol{y}}) + \bar{\boldsymbol{y}}\boldsymbol{b} = \boldsymbol{cx}_j + \bar{\boldsymbol{y}}\boldsymbol{g}_j - \varphi(\bar{\boldsymbol{y}})$$

so that, rewriting (L_β) as

(L_β) $\max_{y,v} \{ v : v \le (\boldsymbol{c} - \boldsymbol{yA})\boldsymbol{x}_j + \boldsymbol{yb} - \varphi(\bar{\boldsymbol{y}}) \ j \in \beta \} + \varphi(\bar{\boldsymbol{y}})$

the above relations on the α_i, the definition $\boldsymbol{y} = \bar{\boldsymbol{y}} + \boldsymbol{d}$ and some simple algebra yield

(L_β) $\max_{y,v} \{ v : v \le \boldsymbol{dg}_j + \alpha_j \ j \in \beta \} + \varphi(\bar{\boldsymbol{y}})$

i.e. (L_β) and (D_β) are exactly the same problem modulo a proper translation w.r.t. the current point $\bar{\boldsymbol{y}}$, and (M_β) can be rewritten as

(M_β) $\min_\theta \{ \alpha\theta : \Sigma_{j \in \beta} \boldsymbol{g}_j\theta_j = \boldsymbol{0} , \theta \in \Theta \}$ [$+ \varphi(\bar{\boldsymbol{y}})$].

If (M_β) has a nonempty feasible region, any feasible solution θ gives an upper bound $\sigma = \alpha\theta$ [$+ \varphi(\bar{\boldsymbol{y}})$] to the optimal value $\varphi(\boldsymbol{y}^*)$: therefore, (M_β) can be seen from the dual viewpoint as the problem of finding the best of such bounds. However, the relations between primal and dual optimality can be made more precise by means of the "sensitivity" (concave and nondifferentiable) function

$$\psi(\gamma) = \min_{\boldsymbol{x}} \{ (\boldsymbol{c} - \gamma)\boldsymbol{x} : \boldsymbol{x} \in X \} = z((P_{c-\gamma}))$$

such that $\varphi(\boldsymbol{y}) = \psi(\boldsymbol{yA}) + \boldsymbol{yb}$: the following theorem gives a "dual flavour" even to the vectors $\boldsymbol{x}$ of the primal space, showing how [ε-]optimality for $(P_{c-\gamma})$ is equivalent to the condition of being an [ε-]subgradient for $\psi()$.

Theorem I.1.2.1: *$\boldsymbol{x}$ is an ε-optimal solution for $(P_{c-\gamma})$ $\Leftrightarrow$ $-\boldsymbol{x}$ is an ε-subgradient for $\psi()$ in γ.*

Proof: [$\Rightarrow$] By ε-optimality of $\boldsymbol{x}$, $\psi(\gamma) + \varepsilon \ge (\boldsymbol{c} - \gamma)\boldsymbol{x}$: clearly, for any γ', $\boldsymbol{x}$ is in general suboptimal, i.e. $(\boldsymbol{c} - \gamma')\boldsymbol{x} \ge \psi(\gamma')$, hence

$$\psi(\gamma') \le (\boldsymbol{c} - \gamma')\boldsymbol{x} + \gamma\boldsymbol{x} - \gamma\boldsymbol{x} = (\boldsymbol{c} - \gamma)\boldsymbol{x} + (\gamma - \gamma')\boldsymbol{x} \le \psi(\gamma) + (\gamma' - \gamma)(-\boldsymbol{x}) + \varepsilon.$$

[$\Leftarrow$] Conversely, suppose $\forall \gamma' \ \psi(\gamma') \le \psi(\gamma) + (\gamma' - \gamma)(-\boldsymbol{x}) + \varepsilon$ holds, but $\psi(\gamma) < (\boldsymbol{c} - \gamma)\boldsymbol{x} - \varepsilon$ (i.e. $\boldsymbol{x}$ is not ε-optimal): then, by taking $\gamma' = \boldsymbol{c}$ in the previous relation, we get

$$0 = \psi(\boldsymbol{c}) \le \psi(\gamma) + (\boldsymbol{c} - \gamma)(-\boldsymbol{x}) + \varepsilon < [(\boldsymbol{c} - \gamma)\boldsymbol{x} - \varepsilon] - (\boldsymbol{c} - \gamma)\boldsymbol{x} + \varepsilon = 0. \qquad \blacklozenge$$

The linearization error $\alpha_i = \varphi(\boldsymbol{y}_i) + \boldsymbol{g}_i(\bar{\boldsymbol{y}} - \boldsymbol{y}_i) - \varphi(\bar{\boldsymbol{y}})$ of the i-th subgradient $\boldsymbol{g}_i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i$ w.r.t. the current point $\bar{\boldsymbol{y}}$ is

$$\begin{aligned} \alpha_i &= \varphi(\boldsymbol{y}_i) + \boldsymbol{g}_i(\bar{\boldsymbol{y}} - \boldsymbol{y}_i) - \varphi(\bar{\boldsymbol{y}}) = \\ &= [\psi(\boldsymbol{y}_t\boldsymbol{A}) + \boldsymbol{y}_t\boldsymbol{b}] + (\bar{\boldsymbol{y}} - \boldsymbol{y}_i)(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i) - [\psi(\bar{\boldsymbol{y}}\boldsymbol{A}) + \bar{\boldsymbol{y}}\boldsymbol{b}] = \\ &= \psi(\boldsymbol{y}_t\boldsymbol{A}) + (-\boldsymbol{x}_i)(\bar{\boldsymbol{y}}\boldsymbol{A} - \boldsymbol{y}_t\boldsymbol{A}) - \psi(\bar{\boldsymbol{y}}\boldsymbol{A}) = \psi(\gamma_i) + (\bar{\gamma} - \gamma_i)(-\boldsymbol{x}_i) - \psi(\bar{\gamma}) \end{aligned}$$

(where $\bar{\gamma} = \bar{\boldsymbol{y}}\boldsymbol{A}$ and $\gamma_i = \boldsymbol{y}_t\boldsymbol{A}$), i.e. it is as well the linearization error of $-\boldsymbol{x}_i$ (as a subgradient of $\psi()$ in γ_i) w.r.t. $\bar{\gamma}$: in other words, $\boldsymbol{x}_i$ is an α_i-*optimal solution* for the pricing subproblem $(P_{c-\bar{\gamma}})$ corresponding to the current point $\bar{\boldsymbol{y}}$. This immediately implies that, for any $\theta \in \Theta$, $\bar{\boldsymbol{x}} = \Sigma_{j \in \beta} \boldsymbol{x}_j\theta_j$ is a σ-subgradient of $\psi()$ in $\bar{\gamma}$, i.e. that it is σ-*optimal for* $(P_{c-\bar{\gamma}})$: hence, it is possible to prove in a much simpler and more general form two theorems that have been proposed [Me94] to measure the "quality" of the current point $\bar{\boldsymbol{y}}$.

Lemma I.1.2.2: *let* $\boldsymbol{x}^*$ *be the optimal solution for (M) and* $\boldsymbol{z} = \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}} = \Sigma_{i \in \beta} \boldsymbol{g}_i \theta_i$ *(for any* $\theta \in \Theta$*): hence,* $\boldsymbol{c}\boldsymbol{x}^* \geq \boldsymbol{c}\bar{\boldsymbol{x}} + \bar{\boldsymbol{y}}\boldsymbol{z} - \sigma$.

Proof: σ-optimality of $\bar{\boldsymbol{x}}$ implies $(\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\boldsymbol{x}^* + \sigma \geq (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\bar{\boldsymbol{x}}$, i.e.

$$\boldsymbol{c}\boldsymbol{x}^* \geq \boldsymbol{c}\bar{\boldsymbol{x}} + \bar{\boldsymbol{y}}(\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{A}\bar{\boldsymbol{x}}) - \sigma$$

but $\boldsymbol{x}^*$ is feasible, i.e. $\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b}$, and this completes the proof. ♦

Lemma I.1.2.3: *for any* $\theta \in \Theta$, $\boldsymbol{c}\bar{\boldsymbol{x}} - \varphi(\bar{\boldsymbol{y}}) \leq \sigma - \bar{\boldsymbol{y}}\boldsymbol{z}$.

Proof: By σ-optimality of $\bar{\boldsymbol{x}}$, $\varphi(\bar{\boldsymbol{y}}) = \psi(\bar{\boldsymbol{y}}\boldsymbol{A}) + \bar{\boldsymbol{y}}\boldsymbol{b} \geq (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\bar{\boldsymbol{x}} - \sigma + \bar{\boldsymbol{y}}\boldsymbol{b}$. ♦

However, the above theorems are of little (if any) use in practice, and the stopping rules discussed in §I.1.1 are typically used instead; since $\boldsymbol{z} = \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}$ is σ-optimal for subproblem $(P_{\boldsymbol{c}-\bar{\boldsymbol{y}}\boldsymbol{A}})$, these rules are a generalization of the stopping condition of DW algorithm, since they hold when a point $\bar{\boldsymbol{x}}$ is found such that constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ are *slightly violated* ($\| \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}} \|^2 \leq \varepsilon_Q$) and $\bar{\boldsymbol{x}}$ is *almost* optimal for the current pricing subproblem. This finally gives a clear interpretation to ε_Q as a measure of the maximum violation of the $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ constraint that can be accepted in the final *primal* solution reported by the algorithm: however, when the stress is more on the *dual* side, i.e. on the determination of the optimal value $\varphi(\boldsymbol{y}^*)$, this interpretation is not of much use in guessing a correct ε_Q, and the alternative stopping rule presented at the end of §I.1.1 can be more effective.

A *primal interpretation* of $(\Delta_{\beta t})$ can be obtained from

$$(M_\beta^\varepsilon) \qquad \min_\theta \{ \alpha\theta : \| \boldsymbol{G}_\beta\theta \|^2 \leq \varepsilon , \theta \in \Theta \},$$

where $\boldsymbol{G}_\beta$ is the matrix having $\{ \boldsymbol{g}_i : i \in \beta \}$ as columns, which is the nonlinear relaxation of (M_β) where the constraints $\boldsymbol{A}(\Sigma_{j \in \beta} \boldsymbol{x}_j\theta_j) = \boldsymbol{b}$ are relaxed in a *Least-Square fashion* to $\| \boldsymbol{A}(\Sigma_{i \in \beta} \boldsymbol{x}_i\theta_i) - \boldsymbol{b} \|^2 \leq \varepsilon$: $(\Delta_{\beta t})$ is a Lagrangean relaxation (with Lagrangean multiplier t) of (M_β^ε). Obviously, (M_β^0) is equivalent to (M_β), since their objective functions and feasible regions are exactly the same: in general, $\{ (M_\beta^\varepsilon) \}$ is a family of relaxations of (M_β) that gets weaker as ε increases. Even from a practical point of view, using (M_β^ε) instead of (M_β) may be reasonable, since it means extending the linear (affine) constraint region $\{ \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \}$ to a cylinder of ray $(2\varepsilon)^{1/2}$ around it, and it is always possible to find a large enough ε which makes (M_β^ε) feasible for every nonempty β (Figure 8), while (M_β) can stay unbounded until a large number of vertices is added: in this sense, the former can be viewed as a "stabilized" version of the latter, that might help in enhancing the poor practical performances of the classical DW method, caused by (the primal interpretation of) the instability problems of the CP algorithm.

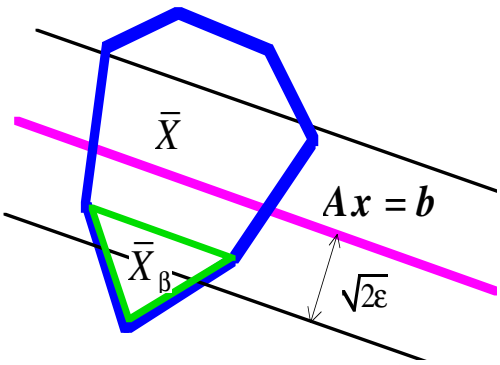


Figure 8: (M_β) can be empty while (M_β^ε) it is not

It is important to remark that the *same* problem $(\Delta_{\beta t})$ can be seen as a Lagrangean relaxation of the *two different* problems (M_β^ε) and $(\Delta_\beta^\varepsilon)$, with Lagrange multiplier t in the first case and $(1 / t)$ in the second: the constraint relaxed from $(\Delta_\beta^\varepsilon)$ is aimed to ensure that all the information used to compute the

direction is reliable, i.e. (almost) in the ε-subdifferential, while the one relaxed from (M_β^ε) is aimed to ensure (almost) feasibility, so that it is possible to see t as a "knob" that can be tuned to privilege either the dual aspect (searching for an ascent direction) or the primal one (searching for a feasible primal solution). This latter interpretation of t, that will be used later to justify new t-strategies in Chapter I.5, can be "verified" by referring to both the viewpoints: from the dual side, t switches from a "local" to a "global" view of the function, that is if $\boldsymbol{d}$ has to be an ascent direction in a small neighbourhood of $\bar{\boldsymbol{y}}$, or rather if $\boldsymbol{d}$ has to point towards the maximum of $\varphi_\beta()$. However, finding a feasible $\bar{\boldsymbol{x}}$ means in fact finding a null subgradient in the σ-subdifferential of the current point, i.e. *demonstrating global* σ*-optimality*: there is a clear difference (typical of many NDO methods) between *reaching* an ε-optimal solution and *demonstrating* its ε-optimality, but both conditions are required to stop, so that t can be seen as a "relative weight" given to the decrease of $\| \boldsymbol{G}_\beta \theta \|^2$ w.r.t. the increase of $\varphi(\bar{\boldsymbol{y}})$. Furthermore, the role of t can be interpreted in exactly the same way by only using primal concepts, which can be clearly seen by considering that in practical DW codes the computation is usually divided in a *Phase* 0 and a *Phase* 1, where the first apply the DW technique to the feasibility (linear) problem

$$(M^p) \qquad \min_\theta \{ \| \boldsymbol{A}(\Sigma_{j \in \beta} \boldsymbol{x}_j \theta_j) - \boldsymbol{b} \|_p : \theta \in \Theta \} \qquad p \in \{1, \infty\}$$

in order to generate a nonempty (M_β), and the second actually optimizes the objective function $\boldsymbol{c}\boldsymbol{x}$ (= $\alpha\theta$); if using $p = 2$ instead, t in ($\Delta_{\beta t}$) can be interpreted as a weight given to the "Phase 0 portion" of a composite objective function trying to balance between the two different requirements.

The above observation suggests the possibility of a "purely primal" Bundle algorithm, closely resembling the DW approach, that be constructed using the (QP) subproblem ($\Delta_{\beta t}$) rather than the (LP) subproblem (M_β): it would not make any use of the typical concepts (such as the Null Step - Serious Step) of "dual" Bundle methods, and even completely disregards the value of the (dual) function $\varphi()$. Although such an approach is probably inefficient in practice, it is a nice illustration of how the conceptual machinery of Bundle methods can be used from a completely different point of view: furthermore, some ideas introduced in the discussion will be useful later on. The basic observation is that any (efficient) algorithm that solves ($\Delta_{\beta t}$) can be used to approach the quadratically constrained problem (M_β^ε): in fact, being a pointwise minimum of an infinite set of linear functions, the univariate function $\delta_\beta(t) = z(\Delta_{\beta t})$ is concave, so that the Lagrangean Dual

$$(T_\beta^\varepsilon) \qquad \max_{t \geq 0} \{ \delta_\beta(t) - \varepsilon t \}$$

can be solved by means of a suitable line search procedure on t; (T_β^ε) is also well known to be equivalent to (M_β^ε), so that the former has finite optimum if and only if the latter is nonempty. More in detail, the nonnegative halfline $\mathcal{R}_+$ can be subdivided into a finite number of intervals $[\, t_j \,,\, t_{j+1} \,]$ (each one being the stability interval of an optimal *base* of ($\Delta_{\beta t}$), Cf. §I.3.6) such that for each $t \in [\, t_j \,,\, t_{j+1} \,]$ the optimal solutions of ($\Delta_{\beta t}$) and ($\Pi_{\beta t}$) are completely described by three numbers $s_j \geq 0$, q_j and $r_j \geq 0$ and two vectors θ_j' and θ_j'', in the sense that $\theta(t) = \theta_j' + (1/t)\theta_j''$ is the optimal solution of ($\Delta_{\beta t}$) with

$$\| \boldsymbol{G}_\beta \theta(t) \|^2 = \| z(t) \|^2 = s_j + (1/t)^2 r_j \qquad \text{and} \qquad \alpha\theta(t) = q_j - (1/t) r_j \qquad (1)$$

while for ($\Pi_{\beta t}$)

$$\| \boldsymbol{d}(t) \|^2 = \| tz(t) \|^2 = t^2 s_j + r_j \qquad \text{and} \qquad v(t) = t s_j + q_j \qquad (2)$$

so that

$$\delta_j(t) = v(t) - {}^1/_2 t \left\| \boldsymbol{d}(t) \right\|^2 = {}^1/_2 [\, t s_j + 2q_j - (1/t) r_j \,] = ({}^1/_2) t \left\| \boldsymbol{G}_\beta \theta(t) \right\|^2 + \alpha\theta(t) \qquad (3)$$

is equal to $\delta_\beta(t)$ for each $t \in [\, t_j , t_{j+1} \,]$. Being the sum of the linear function $({}^1/_2) t s_j + 2q_j$ and of the concave function $-({}^1/_{2t}) r_j$, $\delta_j(t)$ is a (strictly for $r_j > 0$) smooth concave upper approximation of $\delta_\beta(t)$, while the latter is a piecewise concave possibly nonsmooth function (see Figure 9): therefore

$$\delta_j'(t) = {}^1/_2 (s_j + (1/t)^2 r_j) = {}^1/_2 \left\| z(t) \right\|^2 \geq 0$$

is the derivative of $\delta_\beta()$ for any $t \in (\, t_j , t_{j+1} \,)$ and its left and right derivative in the (resp. right and left) extremes of the interval, so that $\delta_\beta()$ is a non decreasing function. If $\delta_\beta()$ is differentiable in the optimal point $\bar{t}$, then $\left\| z(\bar{t}) \right\|^2 = 2\varepsilon$; otherwise, $\bar{t} = t_i$ for some i, and two equivalent optimal solutions $\theta' = \theta_j(\bar{t})$ and $\theta'' = \theta_{j+1}(\bar{t})$ of $(\Delta_{\beta\bar{t}})$ exist such that

$$\varepsilon \geq {}^1/_2 \left\| \boldsymbol{G}_\beta \theta' \right\|^2 \quad \text{and} \quad {}^1/_2 \left\| \boldsymbol{G}_\beta \theta'' \right\|^2 \leq \varepsilon$$

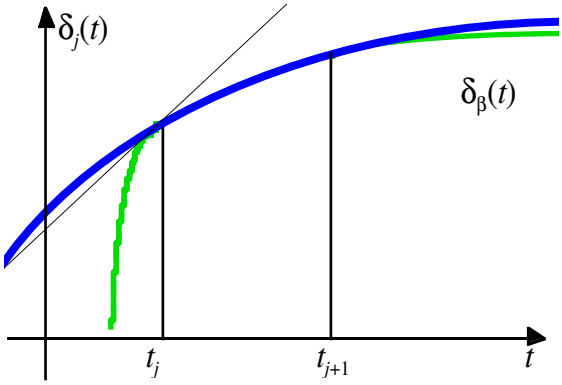


Figure 9: $\delta^+(t)$ is a smooth concave upper approximation of the nonsmooth $\delta_\beta(t)$

where at least one of the two inequalities strictly holds: hence, $\mu\theta' + (1-\mu)\theta''$ is still an optimal solution of $(\Delta_{\beta\bar{t}})$ for each $\mu \in [0, 1]$, and it is easy to find a $\bar{\mu}$ such that

$$\left\| \boldsymbol{G}_\beta [\, \bar{\mu}\theta' + (1-\bar{\mu})\theta'' \,] \right\|^2 = \left\| \boldsymbol{G}_\beta \theta(\bar{t}) \right\|^2 = \left\| z(\bar{t}) \right\|^2 = 2\varepsilon$$

(as $\mu \left\| \theta' \right\|^2 + (1-\mu) \left\| \theta'' \right\|^2 \geq \left\| \mu\theta' + (1-\mu)\theta'' \right\|^2$ for each μ). In other words, $\bar{t}$ is the smallest value of t, if any, for which an optimal solution of $(\Delta_{\beta t})$ exists (and can be constructively found) that is feasible for (M_β^ε): conversely, the existence of an $\bar{\boldsymbol{x}} \in \bar{X}_\beta$ such that $\left\| \boldsymbol{A}\bar{\boldsymbol{x}} - \boldsymbol{b} \right\|^2 \leq \varepsilon$ is equivalent to the existence of a $\bar{t}$ such that the derivative of $\delta_\beta(t) - \varepsilon t$ in $\bar{t}$ is ≤ 0. All the above facts lead to the definition of a Line Search procedure that can be sketched as follows: for given ε, β and t, $(\Delta_{\beta t})$ is solved and its sensitivity analysis parameters are used to find the new t or to detect unboundedness as

- if $s_j < \varepsilon$, then $t' = (r_j / (\varepsilon - s_j))^{1/2}$ is a good value to test: if $t' \in [\, t_j , t_{j+1} \,]$ then it is optimal and explicit formulae for obtaining the optimal θ, v ... exist, otherwise a new interval is visited (the piecewise nature of the function clearly ensures finite termination);
- if $s_j \geq \varepsilon$, then $\left\| z(t) \right\|^2 > 2\varepsilon$ for each $t \in [\, t_j , t_{j+1} \,]$ and some $t' > t_{j+1}$ must be tried in order to see whether a suitable interval exists: if $t_{j+1} = +\infty$ than (T_β^ε) is unbounded, and the current value of t and θ can be used in order to obtain new information.

Hence, in a finite number of steps either the optimal $\bar{t}$ (and the associated $\theta(\bar{t})$, $v(\bar{t})$ ...) is found, or unboundedness is detected; interestingly enough, this is *not* a LS in the space of the dual variables $\boldsymbol{y}$, since $\varphi()$ is *not* evaluated on the corresponding dual points $\{ \boldsymbol{y}_h \}$ (the interest is rather in what the current model says), although one could imagine a procedure that does: moreover, the points $\boldsymbol{y}_h$ do not lie along the same line, so that it would eventually result in a *Curve-Search* approach analogous to the Bundle Trust-Region algorithm of §I.5.2.

A possible algorithm is stated in Figure 10, and its proof of convergence is simple and can be carried

along without ever mentioning the "dual" side. When the new $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{y})$ gives no new information, i.e. $v \le \alpha + \boldsymbol{g}\boldsymbol{y}$, the current value of $\delta_\beta(t)$ is equal to $\delta(t)$ [$= z(\Delta_t)$], since $(v, \boldsymbol{y})$ solves (Π_t): in fact, being $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{y})$ the solution of the pricing subproblem, one has

$$(\boldsymbol{c} - \boldsymbol{y}\boldsymbol{A})\boldsymbol{x} \le (\boldsymbol{c} - \boldsymbol{y}\boldsymbol{A})\boldsymbol{x}' \qquad \forall\, \boldsymbol{x}' \in X \supseteq \bar{X}$$

i.e. $$v \le \alpha + \boldsymbol{y}\boldsymbol{g} = \boldsymbol{c}\boldsymbol{x} + \boldsymbol{y}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) \le \boldsymbol{c}\boldsymbol{x}' + \boldsymbol{y}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}') = \alpha' + \boldsymbol{y}\boldsymbol{g}' \quad \forall\, \boldsymbol{g}'$$

so that $(v, \boldsymbol{y})$ is feasible for (Π_t), $\theta(t)$ is feasible for (Δ_t) and complementary slackness hold.

Stabilized Dantzig-Wolfe

```
t = t0; ε = ε0; β = { g( 0 ) };                    /* initialization */
do
   ( Bounded , t , v, y ) = SolveT( ε , β , t );   /* the above line search on t */
   g = g( y ); α = cx( y );                        /* solve the pricing subproblem */
   if( v > α + gy )                                /* "new information" obtained */
         β = β ∪ { g };                            /* exploit it (enlarge X̄β) */
   else  if( Bounded )                             /* (Mβε) equivalent to (Mε) */
               ε = sj + ( 1 / tj+1 )²rj;           /* increase the required primal accuracy */
         else  t = 2t;
while( ε > 0 );
```

Figure 10: a "stabilized" Dantzig-Wolfe-type algorithm

Therefore, provided that the original problem (P) has a nonempty feasible region, for any fixed ε a "Bounded" condition must be obtained in a finite number of iterations: in fact, at each step either β is enlarged or t is augmented, but there cannot be infinitely many consecutive t augments as this would imply that $\delta()$ goes to infinity (contradicting the boundedness of (T^0)), and for any $\varepsilon \ge 0$ a finite β exists such that (M^ε_β) is nonempty and (T^ε_β) is bounded. The latter argument also shows that there cannot be infinitely many ε decreases, since when β contains enough vertices it exists a t such that $\| z(t) \|^2 = s_j + (1/t)^2 r_j = 0$ ($\Rightarrow s_j = r_j = 0$) that must be at some point reached (as the number of intervals is finite and does not depend on ε) forcing ε to become 0; some subtle details, as how t_{j+1} has to be defined when t is a kink point of $\delta_\beta()$, can be omitted. Now, if the current t solves (T^ε_β), and $v \le \alpha + \boldsymbol{y}\boldsymbol{g}$, then $\theta(t)$ is optimal for the whole (M^ε), since it solves $(\Delta_{\bar t})$ and satisfies $\| \boldsymbol{G}_\beta \theta(\bar t) \|^2 = 2\varepsilon$, so that it also solves (T^ε), that is known to be equivalent to (M^ε): therefore, when the algorithm stops (with $\varepsilon = 0$) the obtained solution is optimal to (M^0), hence to (P).

Note that several questions of relevant importance in practice have not been addressed, such as the "β-strategy" (how and when to eliminate items from β in order to keep it to a reasonable size) and the possibility of changing the current point from $\mathbf{0}$ to any general $\bar{\boldsymbol{y}}$, that can be obviously embedded in this framework: the interest was in making apparent the deep relations between Bundle and Dantzig-Wolfe methods, and to clarify the primal role of the subproblem $(\Delta_{\beta t})$. In the next Chapter, however, some of ideas introduced above will be used in the convergence proof of a very general dual Bundle algorithm.

I.2 Convergence Properties

In this Chapter, an unified convergence analysis will be developed to cover a large family of Bundle algorithms using direction finding subproblems ($\Delta_{\beta t}$): the basic algorithm, quite similar to the "generic" one presented in §I.1.1, is described in Figure 11.

Bundle Algorithm

⟨ let $t^* > 0$ and $\varepsilon > 0$ be fixed; choose the initial $\bar{y}$, t and β ⟩;
do
⟨ solve ($\Pi_{\beta t}$) and ($\Delta_{\beta t}$) for (v , $\boldsymbol{d}$) and (σ , $\| z \|$) ⟩; /* find a direction */
⟨ move along $\boldsymbol{d}$, generating some new $\boldsymbol{g}_i$ and a trial point $\boldsymbol{y}$ ⟩;
if(a large enough improvement has been obtained)
then $\bar{y} = y$; /* a Serious Step */
⟨ update β, (possibly) including some new $\boldsymbol{g}_i$, deleting old ones and/or
adding convex combination of the present ones (typically z) ⟩; /* the β-strategy */
⟨ update t, depending on the previous iterations ⟩; /* the t-strategy */
while($\sigma + t^* \| z \|^2 > \varepsilon$);

Figure 11: the Bundle Algorithm

Despite its generality, it does not cover all the Bundle-like methods proposed in the literature, most notably

- Bundle methods for nonconvex nondifferentiable optimization [Ki91b], or for problems with "complicated" constraints [Ki91a];
- methods that try to extract some sort of second order information about the curvature of the function, for instance by using a variable matrix-induced norm $\|\cdot\|_M$ for the "stabilizing" term of ($\Pi_{\beta t}$) [Mi82] [Mi91] [GM92] [BGL95] [FQ96] [LS96a] [LS96b] [LS96c] [Mi96] or in some different ways [GM91]: while it seems to be hardly possible to define a proper concept of curvature for polyhedral functions, complicating the structure of the quadratic subproblems may hide some relevant detail of our interpretation;
- methods that use different forms of direction finding subproblems, like Proximal Level algorithms [BKL95] [Ki95] [LNN95].

The proof technique adopted in this paragraph is quite standard: convergence is shown in two steps, making it apparent the "two-level" nature of these algorithms, where sequences of NS and SS "aims"

to different targets; hence, after some preliminaries presented in §I.2.1, the two §I.2.2 and §I.2.3 will be devoted to the separate analysis of the two "levels" of the proof. The main original contribution of our analysis is that, rather than proving the convergence of a single particular algorithm, it focuses on the general properties that any Bundle approach must have: moreover, the proof techniques have been chosen in order to be easily extendible to the new family of (LP)-based Bundle methods described in Chapter I.4. Such an extension might be possible for Proximal Level or Proximal Point algorithms also, and it is currently under research.

I.2.1 Preliminaries and Convergence Conditions

It is possible to prove convergence of any algorithm fitting in the above framework at the only condition that the following five general properties are respected:

(i) $\bar{t} \geq t_i \geq \underline{t} > 0$ for all iterations i and fixed $\bar{t}$, $\underline{t}$.

(ii) If a SS is performed moving to $\boldsymbol{y}$, then $\varphi(\boldsymbol{y}) - \varphi(\bar{\boldsymbol{y}}) \geq m_1 v$ for a fixed $m_1 \in (0, 1)$: the converse is *not* required, in the sense that even when a "considerable" is obtained, there is the choice of not accepting the step, but for what is required by *(iv)* below.

(iii) In case that the algorithm performs an *infinite* sequence of *consecutive* NS, t has to be *finitely convergent*, i.e. for some h it must be $t_i = t_h > 0$ for all $i \geq h$.

(iv) In the case of *(iii)*, at any iteration $i \geq h$ where $\varphi(\boldsymbol{y}_i) - \varphi(\bar{\boldsymbol{y}}_i) \geq m_1 v$ a SS must be performed; in other words, after *finitely many* NS *any* step that improves at least of $m_1 v_i$ must be accepted.

(v) In the case of *(iii)*, at any iteration $i \geq h$ the point $\bar{\boldsymbol{y}}_i + \boldsymbol{d}_i$ must always be among those that are probed by the line search; furthermore, the corresponding subgradient must be added to β.

A brief initial discussion of the above conditions is in order: *(i)* is necessary only for the proof of convergence in the limit, but it also simplifies somewhat the proofs, and its possible relaxations will be discussed each time it will be used. Although from a theoretical viewpoint *(i)* is a rather strong condition, in practice some "external" knowledge about the proper range of values for t should be given to the algorithm anyway, so that it is not difficult to choose a small $\underline{t}$ and a large $\bar{t}$ that does not interfere with the t-strategy; besides, $\bar{t}$ and $\underline{t}$ *are* ultimately fixed by the finite machine precision of any reasonable implementation. The other conditions just describe the basic ideas of the algorithm: at each step, either a "consistent" increase is obtained by performing a SS, or new information enriches the Bundle; to gain flexibility, however, the algorithm is allowed to "to resist the temptation" of moving to a better point if increasing t seems to be preferable *(ii)*, but it is obviously required that this does not happen forever *(iv)*. Analogously, conditions *(iii)* and *(v)* allow for trying to adjust t in response to a "bad" step, but prevent this adjustment phase to impair the overall convergence of the algorithm. Note that the first Bundle methods proposed in the literature used ad-hoc Line Search procedures to ensure that, in case of a NS, the new $(\boldsymbol{g}_i, \alpha_i)$ gives "reliable" information, i.e. that α_i is "small": all the recent methods are designed to visit $\bar{\boldsymbol{y}} + \boldsymbol{d}$ only, since this is generally preferable in terms of performances, but a LS may still be useful under some circumstances, and the convergence analysis is just not affected at all if *(v)* holds.

An important detail of any implementable Bundle method is the management of β (β-*strategy*): in general, at any step the following two possibilities are given for changing β

- some $\boldsymbol{g} \in \beta$ is removed and / or
- given some $\theta \in \Theta$ [usually the latest optimal solution of $(\Delta_{\beta t})$],
  $(\boldsymbol{g}', \alpha') = (\boldsymbol{G}_\beta\theta, \alpha_\beta\theta)$ [$= (\boldsymbol{z}, \sigma)$] is added to β.

Each of these "moves" may, both in theory and in practice, impair finite termination, or even convergence, of the algorithm: hence, other than the rules *(i)* - *(v)*, a (theoretically) convergent method must also respect some restriction on the β-strategy, that will be extensively discussed in the following. The development of suitable assumptions on the β-strategy is a crucial part of the convergence proofs: in fact, this is perhaps the point where the results of this Chapter differ more from the extensions of Chapter I.4.

In the convergence proof, it will be important to distinguish whether, for a given β, the Cutting Plane model $\varphi_\beta()$ is an upper bounded function (briefly, is bounded) or not: the following observation will be repeatedly useful.

Observation I.2.1.1: $\varphi_\beta()$ *is bounded* $\Leftrightarrow \exists\ \theta \in \Theta$ *such that* $\boldsymbol{G}_\beta\theta = \boldsymbol{0}$.

Proof: $\boldsymbol{G}_\beta\theta = \boldsymbol{0}$ ensures that (M_β) has a feasible solution, therefore its dual (L_β) (equivalent to the maximization of $\varphi_\beta()$) cannot be unbounded; conversely, if $\varphi_\beta()$ is bounded then for some sufficiently large ε ($\geq \max_{j \in \beta} \{ \alpha_j \}$) one has $\boldsymbol{0} \in \partial_\varepsilon\varphi(\bar{\boldsymbol{y}})$. ♦

The first thing to prove is that if the algorithm stops, then the provided solution is "satisfactory": in this general setting, this statement takes the form

Theorem I.2.1.2: *if the stopping condition* $\sigma + t^* \| z \|^2 \leq \varepsilon$ *holds, and* $\| \boldsymbol{G}\theta^* \| = 0$ *for any optimal solution* θ^* *of* (Δ_{t^*})*, then* $\varphi(\boldsymbol{y}^*) - \varphi(\bar{\boldsymbol{y}}) = z(\Delta_{t^*}) \leq \varepsilon$.

Proof: First of all, note that θ^* is required to be an optimal solution of (Δ_{t^*}), i.e. perfect knowledge of the problem is assumed in the "neighbourhood" of $\bar{\boldsymbol{y}}$ determined by t^*: for polyhedral functions, this condition can actually be finitely checked. By the formulae (1), (2) and (3) in §I.1.2, $\| \boldsymbol{G}\theta^* \| = 0$ implies $v^* = \varphi(\bar{\boldsymbol{y}} + \boldsymbol{d}^*) = z(\Pi_{t^*}) = z(\Delta_{t^*}) = \alpha\theta^*$; furthermore, in this case θ^* is clearly an optimal solution to (Δ_t) for any $t \geq t^*$, since $z(\Delta_t) \geq z(\Delta_{t^*})$ (being $t\| \boldsymbol{G}\theta \|^2 + \alpha\theta$ a nondecreasing function in t). But $z(\Delta_t) \leq t \cdot 0 + \alpha\theta^* = z(\Delta_{t^*})$, since θ^* is a feasible solution for (Δ_t): this just means that the CP model is bounded, so that $\varphi(\bar{\boldsymbol{y}}) + \varphi_\beta(\boldsymbol{d}^*) = \varphi(\bar{\boldsymbol{y}} + \boldsymbol{d}^*)$ is at the same time an upper bound and a lower bound on $\varphi(\boldsymbol{y}^*)$, hence $\boldsymbol{y}^* = \bar{\boldsymbol{y}} + \boldsymbol{d}^*$ is an optimal solution for (D). Now, since clearly $\sigma + t^* \| z \|^2 \geq z(\Delta_{t^*})$, it is possible to conclude

$$\varphi(\boldsymbol{y}^*) - \varphi(\bar{\boldsymbol{y}}) = \varphi_\beta(\boldsymbol{d}) = z(\Delta_{t^*}) \leq \sigma + t^* \| z \|^2 \leq \varepsilon. \qquad \blacklozenge$$

As usual in Bundle methods, the stopping condition is completely satisfactory only if $\| z \| = 0$ (which has to be true for the particular case of $\varepsilon = 0$), since then the above theorem applies and $\boldsymbol{y}^*$ is guaranteed to be an ε-optimal solution; in all the other cases, no real estimate in the "quality" of the accepted solution $\bar{\boldsymbol{y}}$ is available, even though in practice guessing a value of t^* that produces a "true" ε-optimal solution is fairly easy.

Following the typical proof pattern for this class of algorithms, the rest of convergence proof will be divided into two parts: in the first it is assumed that no Serious Steps occur, i.e. we examine sequences of consecutive Null Steps, so that in the second it is possible to disregard what happens between two successive SSs, essentially reducing a run of the algorithm to its Serious Steps alone. The rationale for this proof structure is that *a SS destroys the local view* of the function on which the NS idea is based, making it hard to "collate the pieces" *before* and *after* a SS: this can be made more precise by referring to the Moreau-Yosida Regularization of $\varphi()$ or to its "primal counterpart", as shown in the next paragraph.

I.2.2 Null Steps convergence

This part of the convergence proof will mainly use the dual view of the algorithm, where $(\Pi_{\beta t})$ is the "restriction" to the CP-model $\varphi_\beta()$ of

$$(\Pi_{\bar{y},t}) \qquad \phi_t(\bar{y}) = \max_d \{ \varphi(\bar{y} + d) - 1/2t \| d \|^2 \}$$

i.e. the Moreau-Yosida Regularization of $\varphi()$ (Cf. §I.1.1): the function $\phi_t()$ has many nice properties [HL93 - §XV.4], such as of being differentiable even if $\varphi()$ is not, and that

$$\mathbf{0} \text{ is optimal for } (\Pi_{\bar{y},t}) \quad \Rightarrow \quad \phi_t(\bar{y}) = \varphi(\bar{y}) \quad \Rightarrow \quad \bar{y} \text{ is optimal for (D)}$$

since $\partial[\, \varphi(\cdot) - 1/2t \|\cdot\|^2 \,](\mathbf{0}) = \partial\varphi(\mathbf{0})$, so that the only way for $\boldsymbol{d} = \mathbf{0}$ to be optimal for $(\Pi_{\bar{y},t})$ is that $\mathbf{0} \in \partial\varphi(\bar{y})$, i.e. $\bar{y}$ be optimal for (D). However, the "primal counterparts" of the some of the arguments will be analysed, both for their intrinsic interest and for setting up the notations for §II.4.2: the main observation is that $(\Delta_{\beta t})$ can be viewed as the *restriction to* $\overline{X}_\beta$ of

$$(P_{\bar{y},t}) \qquad \min_x \{ f_{\bar{y},t}(\boldsymbol{x}) = (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\boldsymbol{x} + 1/2t \| \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \|^2 : \boldsymbol{x} \in \overline{X} \} \; [\, + \bar{\boldsymbol{y}}\boldsymbol{b} \,]$$

that is an *Augmented Lagrangean* relaxation of (P), where a linear penalty term $\bar{\boldsymbol{y}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})$ (with Lagrangean multipliers $\bar{\boldsymbol{y}}$) and a quadratic one $1/2t \| \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \|^2$ (with Lagrangean multiplier t) are added to the original cost function $\boldsymbol{c}\boldsymbol{x}$. Another way of looking at $(P_{\bar{y},t})$ is as the pricing subproblem $(P_{\boldsymbol{c}-\bar{\boldsymbol{y}}\boldsymbol{A}})$ with an extra "stabilizing" quadratic term that makes it more difficult to solve, justifying some sort of *inner linearization* (otherwise called *Simplicial Decomposition* - see II.1.4.2) approach: in fact, this paragraph will be essentially devoted to showing that a *sequence of consecutive NS* is nothing but an inner linearization approach applied to $(P_{\bar{y},t})$ problem. Analogously to what already did for α, in the following the unnecessary references to $\bar{\boldsymbol{y}}$ (that however is fixed during a sequence of NS) will be dropped: hence, until the end of this paragraph the iteration index i will denote the i-th Null Step of the (only) *infinite* sequence of consecutive NS that the algorithm is supposed to perform, so that v_i, t_i, f_i ... will denote the corresponding values of v, t, $f_t(\bar{\boldsymbol{x}})$ ...

If an infinite sequence of NS is performed, the sequence of the optimal objective function values converges to $z(P_t)$ (for some t that may not be known in advance): the proof makes use of a "compactness" result contained in the following observation.

Observation I.2.2.1: *if an infinite sequence of NS is performed, all the "representative"* $\boldsymbol{y}_i$ *generated belong to a proper ball B centered on* $\bar{\boldsymbol{y}}$.

Proof: According to *(v)*, a "representative" point $\boldsymbol{y}_i = \bar{\boldsymbol{y}} + \boldsymbol{d}_i$ is generated at each iteration, and, in view of *(iii)*, it is possible to disregard whatever had happened before iteration h, when t gets fixed (an alternative statement would be that once a ball containing all the $\boldsymbol{y}_i$, $i \geq h$, has been found, it can always be finitely enlarged to contain the finitely many $\boldsymbol{y}_i$ for $i < h$). Now, it is easy to see that

$$1/2t\| z_i \|^2 \leq 1/2t\| z_i \|^2 + \sigma_i \leq 1/2t\| z_h \|^2 + \sigma_h \qquad \forall\, i \geq h$$

but $\boldsymbol{d}_i = t\boldsymbol{z}_i$, and this completes the proof. ♦

Now, the main result can be proved: an infinite sequence of NS makes $z(\Delta_{\beta t})$ to converge to $z(P_t)$ (for the t to which $\{ t_i \}$ finitely converges) even under the "minimal" conceivable form of β-strategy, where at each step the Bundle only contains the *aggregate subgradient* $\boldsymbol{z}$ (that belongs to the σ-subdifferential of $\varphi()$ in $\bar{\boldsymbol{y}}$); note how a Bundle method using such a β-strategy closely resembles some of the extended subgradient methods that have been proposed in the literature [Br95].

Theorem I.2.2.2: *if an infinite sequence of NS is performed,* $m_1 < 1$ *and at each step* β *is replaced with the set* $\{ (\boldsymbol{z}, \sigma) \}$*, then* $\{ v_i \} \to 0$ *and* $\varphi(\bar{\boldsymbol{y}}) = \phi(\bar{\boldsymbol{y}}) = z(P)$.

Proof: As in Observation I.2.2.1, everything that has happened before iteration h - that of *(iii)* - is disregarded, so that $t = t_i = t_h$ and the iteration index $i \geq h$ can be dropped: hence, $(\Delta_{\beta t})$ is reduced to the following one-dimensional (QP)

$$\min_\gamma \left\{ h(\gamma) = 1/2t\| \gamma\boldsymbol{z} + (1-\gamma)\boldsymbol{g} \|^2 + \gamma\sigma + (1-\gamma)\alpha : \gamma \in [0, 1] \right\}$$

whose optimality conditions can be algebraically determined. Some boring calculations (or the help of an algebraic package) show that

$$h'(\gamma) = t(\boldsymbol{z} - \boldsymbol{g})(\gamma(\boldsymbol{z} - \boldsymbol{g}) + \boldsymbol{g}) + \sigma - \alpha$$

so that $h'(1) = (t\| \boldsymbol{z} \|^2 + \sigma) - (t\boldsymbol{z}\boldsymbol{g} + \alpha)$ and $\gamma^* = \dfrac{t\boldsymbol{g}(\boldsymbol{z} - \boldsymbol{g}) + \alpha - \sigma}{t\| \boldsymbol{z} - \boldsymbol{g} \|^2}$

where γ^* is the optimal solution disregarding the constraints $\gamma \in [0, 1]$, hence

$$h(1) - h(\gamma^*) = \frac{(t(\| \boldsymbol{z} \|^2 - \boldsymbol{z}\boldsymbol{g}) + \sigma - \alpha)^2}{t\| \boldsymbol{z} - \boldsymbol{g} \|^2} = \frac{h'(1)^2}{t\| \boldsymbol{z} - \boldsymbol{g} \|^2}.$$

Since *(iv)* guarantees $t\boldsymbol{g}\boldsymbol{z} + \alpha = \varphi(\boldsymbol{y}) - \varphi(\bar{\boldsymbol{y}}) < m_1 v = m_1(t\| \boldsymbol{z} \|^2 + \sigma)$, the derivative of $h()$ in 1 is strictly positive, hence $z(\Delta_{\beta t})$ will strictly decrease: in fact, $h'(1) \geq (1 - m_1)(t\| \boldsymbol{z} \|^2 + \sigma) = (1 - m_1)h(1) = (1 - m_1)z(\Delta_{\beta t})$. If $\gamma^* \geq 0$, this suffices to show that the $z(\Delta_{\beta t})$ will (sublinearly, with squared root rate) converge to zero: hence $\{ v_i \}$ will also do, because Observation I.2.2.1 guarantees that $\boldsymbol{y}$ belongs to a compact set B, hence $t\| \boldsymbol{z} - \boldsymbol{g} \|^2$ is upper bonded ($\varphi()$ is a Lipschitz function). Conversely, if $\gamma^* < 0$ then $h'(0) > 0$, i.e. $\gamma = 0$ is the optimal solution, and since

$$t\boldsymbol{g}\boldsymbol{z} + \alpha < m_1(t\| \boldsymbol{z} \|^2 + \sigma) \quad \text{and} \quad t\boldsymbol{g}\boldsymbol{z} + \sigma > t\| \boldsymbol{g} \|^2 + \alpha$$

just summing up and dividing by 2 gives

$$h(0) = 1/2t\| \boldsymbol{g} \|^2 + \alpha < 1/2m_1 t\| \boldsymbol{z} \|^2 + \frac{1 + m_1}{2}\sigma$$

that (since $m_1 \leq (1 + m_1) / 2 \leq 1$ for each $m_1 \in [0, 1]$) finally allows one to conclude

$$h(0) < \frac{1 + m_1}{2}(t\| z \|^2 + \sigma) = \frac{1 + m_1}{2} h(1)$$

i.e. linear convergence rate of $z(\Delta_{\beta t})$ (hence of v) to 0; clearly, this implies that $\boldsymbol{d} = \boldsymbol{0}$ is optimal for (Π_t), i.e. that $\phi(\bar{\boldsymbol{y}}) = \varphi(\bar{\boldsymbol{y}})$ and therefore that $\bar{\boldsymbol{y}}$ is optimal for (D). ♦

A form of this result was first proved in [Ki85a], with the extra assumption (deriving from the particular form of Bundle algorithm used) that $\alpha_i = 0$; another related result can be found in [SZ92], but the actual form of Theorem I.2.2.2 is new.

A primal interpretation easily follows, because the "aggregated gradient" $\boldsymbol{z}$ corresponds to a "current primal point" $\bar{\boldsymbol{x}}$, and it is not hard to see that $(\boldsymbol{c} - (\bar{\boldsymbol{y}} + \boldsymbol{d})\boldsymbol{A})$ is the gradient of $f_t()$ at $\bar{\boldsymbol{x}}$: hence, the pricing subproblem $(P_{\boldsymbol{c}-(\bar{\boldsymbol{y}}+\boldsymbol{d})\boldsymbol{A}})$ is nothing but the *linearization* of (P_t) using as objective function $\nabla f_t(\bar{\boldsymbol{x}})\boldsymbol{x}$, i.e. a first-order model of $f_t()$ at $\bar{\boldsymbol{x}}$. The new "representative" vertex $\boldsymbol{x}$ corresponding to $\boldsymbol{g}$ is the solution of the linearized subproblem, that is used to find a descent direction; in fact, by using

$$\alpha = (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\boldsymbol{x} - \varphi(\bar{\boldsymbol{y}}) + \bar{\boldsymbol{y}}\boldsymbol{b} \Rightarrow \sigma = (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\bar{\boldsymbol{x}} - \varphi(\bar{\boldsymbol{y}}) + \bar{\boldsymbol{y}}\boldsymbol{b} \qquad \text{and}$$

$$\boldsymbol{g} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \Rightarrow \boldsymbol{z} = \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}}$$

it is easy to verify that the univariate (QP) in the proof of Theorem I.2.2.2 is

$$\min_\gamma \{ f_t(\gamma\bar{\boldsymbol{x}} + (1 - \gamma)\boldsymbol{x}) : \gamma \in [0, 1] \}$$

i.e. the exact minimization of $f_t()$ along the direction $\boldsymbol{x} - \bar{\boldsymbol{x}}$.

The convergence proof clearly motivates the definition

Definition I.2.2.3: *a* β*-strategy is* ***monotone*** *if, in the case of (iii), for each* $i \geq h$ *either all the* g_j *such that* $\theta_j > 0$ *are kept in* β*, or* (z_i , σ_i) *is added to* β*.*

It is clear that the thesis of Theorem I.2.2.2 follows *a fortiori* by the fact that a monotone β-strategy is used, since in this case the univariate (QP) in the proof of the Theorem is a restriction of $(\Delta_{\beta t})$.

I.2.3 Serious Steps convergence

In this paragraph, all that happens between two consecutive SS is disregarded: therefore, for ease of notation, v_i, t_i ... will rather denote the values of v, t ... at the i-th Serious Step. A weak finite termination proof, requiring $\varepsilon > 0$, is clearly at hand.

Theorem I.2.3.1: *if* $\varphi()$ *is bounded, then the algorithm finitely terminates for any fixed* $\varepsilon > 0$.

Proof: The above discussion has already ruled out the possibility on an infinite sequence of consecutive NS, ensuring that either a SS is performed, or the stopping condition $\sigma + t^*\| z \|^2 \leq \varepsilon$ is met: then, at the i-th SS a lower bound on v_i is available, since $\sigma_i + t^*\| z_i \|^2 > \varepsilon \Rightarrow v_i \geq \varepsilon$ if $t_i \geq t^*$, while if $t_i < t^*$

$$\sigma_i + t^*\| z_i \|_p > \varepsilon \quad \Rightarrow \quad \| z_i \|_p > (\varepsilon - \sigma_i) / t^* \quad \Rightarrow \quad v_i > t_i [(\varepsilon - \sigma_i) / t^*] + \sigma_i$$

$$\Rightarrow \quad v_i > \varepsilon(t_i / t^*) \quad (t_i / t^* < 1).$$

But *(ii)* implies that $\Delta\varphi_i \geq m_1 v_i \Rightarrow \varphi(\bar{\boldsymbol{y}}_i) \geq \varphi(\boldsymbol{y}_0) + m_1(\Sigma_{j<i} v_j) \geq \varphi(\boldsymbol{y}_0) + (m_1\varepsilon / t^*)\Sigma_{j<i} t_j$, and since *(i)* implies that $\Sigma_{j=1..\infty} t_j = \infty$, if infinitely many SS are performed then $\{ \varphi(\bar{\boldsymbol{y}}_i) \} \rightarrow \infty$. If a finite upper bound on the optimal value of $\varphi()$ were known, it would be possible to terminate in a finite number of steps, detecting unboundedness of (D), i.e. emptiness of the feasible set of (P). ♦

Here, $t_i \geq \underline{t}$ of *(i)* can be relaxed to the "classical" $\Sigma_{i=1..\infty} t_i = \infty$ (involving only the value of t during SS), eventually allowing $\{ t_i \} \rightarrow 0$: also, $\bar{t}$ has never been mentioned yet. Note that the "rate of convergence" that could be desumed from the theorem is adversely affected by a small t: this makes only a little sense, since a too large t may cause a large number of NS to be performed before any SS, but the computational experience seem to show that one of the main practical issues of Bundle algorithms (at least when applied to polyhedral functions) is actually to avoid t getting too small, how we will discuss in details in the next paragraph.

It is now time to turn to the subtler case of convergence in the limit sense, corresponding to a run of the algorithm with $\varepsilon = 0$: the (apparently new) proof presented here is based on an idea of [HL93 Theorem 2.1.5], and has been chosen for "compatibility" with the results of Chapter I.4.

Theorem I.2.3.2: *the sequence* $\{ \varphi(\bar{\boldsymbol{y}}_i) \}$ *converges to a maximizer* φ^* *of* $\varphi()$.

Proof: being a strictly increasing sequence, $\{ \varphi(\bar{\boldsymbol{y}}_i) \}$ converges to some φ^*: if $\varphi^* = \infty$ then it is a maximizer of $\varphi()$ and there is nothing to prove, while if the algorithm finitely terminates then $\varphi^* < \infty$ and the obtained solution is optimal (Theorem I.2.1.2). So, assume that $\varphi^* < \infty$ and the algorithm never stops: it must either perform an infinite sequence of SS or an infinite sequence of consecutive NS, but in the latter case the point $\boldsymbol{y}^*$ reached by the last SS must be optimal (Theorem I.2.2.2).

Hence, assume that infinitely many SS are performed, but that there exist a point $\hat{\boldsymbol{y}}$ and a $\delta > 0$ such that $\varphi(\hat{\boldsymbol{y}}) \geq \varphi(\bar{\boldsymbol{y}}_i) + \delta$ for all i: we claim that, on the limit, $\hat{\boldsymbol{y}}$ must be "farther" from $\boldsymbol{y}_i$ than $\boldsymbol{y}_{i+1}$, i.e. outside the level set $B_i = \{ \boldsymbol{y} : \| \boldsymbol{y} - \bar{\boldsymbol{y}}_i \| = \| \boldsymbol{d}_i \| \}$. Formally, being $\rho_i = \| \boldsymbol{d}_i \|$ the radius of B_i and $\omega_i = \| \boldsymbol{w}_i \| = \| \hat{\boldsymbol{y}} - \bar{\boldsymbol{y}}_i \|$ the distance of $\hat{\boldsymbol{y}}$ from $\bar{\boldsymbol{y}}_i$, the aim is to establish that, for any sufficiently large i, $\omega_i > \rho_i$. As in the proof of Theorem I.2.3.1, it is easy to see that it must be $\Sigma_{j=i..\infty} v_j < \infty$, for otherwise $\varphi^* = \infty$: but ω_i must be bounded away from zero, for otherwise a subsequence of the $\{ \bar{\boldsymbol{y}}_i \}$ would converge to $\hat{\boldsymbol{y}}$ ($\Rightarrow \varphi(\hat{\boldsymbol{y}}) \leq \varphi^*$), while on the contrary ρ_i must converge to zero fast enough, for otherwise $\Sigma_{j=i..\infty} v_j = \infty$ as $v_i \geq t_i \| \boldsymbol{z}_i \|^2 = \| \boldsymbol{d}_i \|^2 / t_i = (\rho_i)^2 / t_i$ and t_i is upper bounded. Now, let $\boldsymbol{r}_i$ be the intersection between the frontier of B_i and $\mathrm{conv}\{ \hat{\boldsymbol{y}}, \bar{\boldsymbol{y}}_i \}$: one has

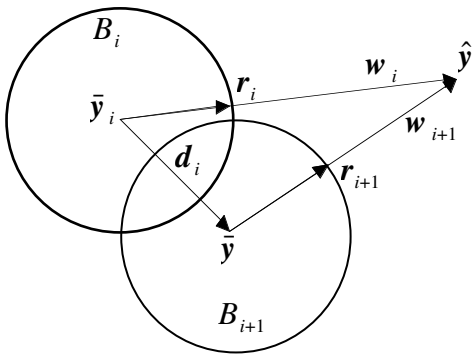


Figure 12: the induction step

- $\| \boldsymbol{r}_i - \bar{\boldsymbol{y}}_i \| = \rho_i < \omega_i$;
- $\varphi(\bar{\boldsymbol{y}}_i) + v_i = \bar{\varphi}_\beta(\bar{\boldsymbol{y}}_{i+1}) \geq \bar{\varphi}_\beta(\boldsymbol{r}_i) \geq \varphi(\boldsymbol{r}_i)$;
- since $\boldsymbol{r}_i = \bar{\boldsymbol{y}}_i + (\rho_i / \omega_i)\boldsymbol{w}_i = (1 - \rho_i / \omega_i)\bar{\boldsymbol{y}}_i + (\rho_i / \omega_i)\hat{\boldsymbol{y}}$ and $\varphi()$ is concave

$$\varphi(\boldsymbol{r}_i) \geq \left(1 - \frac{\rho_i}{\omega_i}\right)\varphi(\bar{\boldsymbol{y}}_i) + \frac{\rho_i}{\omega_i}(\varphi(\bar{\boldsymbol{y}}_i) + \delta) \Rightarrow v_i \geq \delta\frac{\rho_i}{\omega_i}.$$

Reasoning about σ_i as we did for ρ_i, it is easy to show that, for a sufficiently large i, $\boldsymbol{z}_i$ is a σ_i-subgradient of $\varphi()$ in $\bar{\boldsymbol{y}}_i$, hence by the subgradient inequality

$$\boldsymbol{d}_i \boldsymbol{w}_i = t_i \boldsymbol{z}_i (\hat{\boldsymbol{y}} - \bar{\boldsymbol{y}}_i) \geq t_i (\delta - \sigma_i) > t_i (\delta / 2)$$

and this is used twice: the first is to obtain $\rho_i \omega_i > t_i (\delta / 2)$, hence

$$\infty > \Sigma_{j = i..\infty} v_j > \delta \Sigma_{j = i..\infty} \frac{\rho_j}{\omega_j} > \frac{\delta^2}{2} \Sigma_{j = i..\infty} \frac{t_j}{\omega_j^2}$$

and the second is, together with $\boldsymbol{d}_i + \boldsymbol{w}_{i+1} = \boldsymbol{w}_i$ (Figure 12) and the triangle inequality, to obtain

$$\| \boldsymbol{w}_{i+1} \|^2 = \| \boldsymbol{w}_i \|^2 + \| \boldsymbol{d}_i \|^2 - 2 \boldsymbol{w}_i \boldsymbol{d}_i \quad \Rightarrow \quad \omega_{i+1}^2 = \omega_i^2 + \rho_i^2 - t_i \delta < \omega_i^2 + \rho_i^2 .$$

This can be used recursively in the previous relation, together with $t_i \geq \underline{t}$, to obtain

$$\infty > \frac{\delta^2}{2} \Sigma_{j = i..\infty} \frac{t_j}{\omega_j^2} > \frac{\underline{t} \delta^2}{2} \left(\frac{1}{\omega_i^2} + \frac{1}{\omega_i^2 + \rho_i^2} + \frac{1}{\omega_i^2 + \rho_i^2 + \rho_{i+1}^2} + \dots \right) >$$

$$> \underline{t} \delta \Sigma_{j = i..\infty} \frac{1}{\omega_i^2 + (j - i) \bar{\rho}}$$

that yields the desired contradiction, since the rightmost series is divergent[Δ]; in the formula, $\bar{\rho}$ is any fixed number larger than any ρ_j for $j \geq i$, that exists since $\{ \rho_i \} \rightarrow 0$. ♦

The above proof is not very strong - namely, convergence is only shown for the value of the objective function - especially if compared e.g. to that of [CL93]; however, it has the advantage of being easily extendible to the larger class Bundle methods presented in Chapter I.4, which is *not* true for all the "standard" proof in the literature. Furthermore, an even stronger property can be proved for polyhedral functions: in this case, finding a "true" optimal solution (i.e. not just optimal within some $\varepsilon > 0$) in a finite number of steps is possible (while it is in general hopeless if $\varphi()$ is not polyhedral) provided that the "dangerous" subgradient aggregations are properly limited.

Definition I.2.3.3: *a* β*-strategy is* ***safe*** *if*

- *the total number of aggregations* (*adding* $(\boldsymbol{z}_i , \sigma_i)$ *to* β) *is finite and*
- *if some* $\boldsymbol{g}_j$ *has been removed from* β *after a SS, no other removal is permitted unless* $\bar{\varphi}_\beta (\bar{\boldsymbol{y}}_k) > \bar{\varphi}_\beta (\bar{\boldsymbol{y}}_i)$*, where* $k < i$ *is the latest iteration where a removal took place.*

This definition is made by two parts, whose role will be made clear by the proof of the following theorem: the desired result also requires the following very simple observation

---

[Δ] The n-th term of the series is (but for a constant term, proportional to) the *digamma* function $\psi()$ evaluated in n, i.e. the logarithmic derivative $\psi(n) = \Gamma'(n) / \Gamma(n)$ of Euler's $\Gamma()$ function: although unbounded, $\psi()$ grows very slowly, making it difficult to prove convergence with anything but a lower bounded t. For instance, if a t proportional to $1 / n$ is used instead, the corresponding series is not divergent any more, although this is not necessarly true for the series *prior* the minorizations.

Observation I.2.3.4: *let* Γ *be a finite set of vectors* $\{ \boldsymbol{g}_1 \dots \boldsymbol{g}_q \}$*: there exists a* $\mu > 0$ *depending only on* Γ *such that, for any* $\beta \subset \Gamma$,

$$\exists\, \theta \in \Theta \text{ such that } \left\| \boldsymbol{G}_\beta \theta \right\| < \mu \Rightarrow \exists\, \theta' \in \Theta \text{ such that } \boldsymbol{G}_\beta \theta' = \boldsymbol{0}.$$

Proof: Just denote $\mu(\beta) = \min_\theta \{ \left\| \boldsymbol{G}_\beta \theta \right\| : \theta \in \Theta \}$ and let $\underline{\Gamma} = \{ \beta : \mu(\beta) > 0 \} \subseteq \Gamma$: since $\underline{\Gamma}$ is a *finite* set, $\mu = \min\{ \mu(\beta) : \beta \in \underline{\Gamma} \} > 0$, and any β such that $\left\| \boldsymbol{G}_\beta \theta \right\| < \mu$ for some $\theta \in \Theta$ (if any) must belong to $\Gamma / \underline{\Gamma}$ - the fact that $\left\| \boldsymbol{v} \right\| = 0 \Leftrightarrow \boldsymbol{v} = \boldsymbol{0}$ does the rest. ♦

In view of Observation I.2.1.1, this Observation says that if there exists a small-norm vector in the convex hull of the $\boldsymbol{g}_j \in \beta$, then $\varphi_\beta()$ is bounded.

Theorem I.2.3.5: *let* $\varphi()$ *be a polyhedral function: if a safe* β*-strategy is used, the algorithm run with* $\varepsilon = 0$ *will finitely converge to a pair of respectively primal and dual optimal solutions* $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ *if and only if* $\varphi()$ *is bounded.*

Proof: We already know that if $\varphi^* = \infty$ then the algorithm will never stop (unless some extra stopping criteria are used), and if the algorithm stops at some iteration h then it has reached the optimum: hence, suppose that $\varphi^* < \infty$ but the algorithm runs forever. As in the proof of Theorem I.2.3.2, it is easy to obtain that $\forall\, \xi > 0\ \exists\, h$ s.t. $\left\| \boldsymbol{G}_\beta \theta_i \right\| < \xi\ \forall\ i > h$ (for otherwise $\Sigma_{j = 1..\infty}\, v_j = \infty$, since $v_i \geq \underline{t} \left\| z_i \right\|^2$): hence, starting from some step h, the CP-model will always be bounded. In fact, the set Γ_i of all the subgradients generated so far is finite, and Observation I.2.3.4 gives, at each iteration, a lower bound μ_i on the values of $\left\| \boldsymbol{G}_\beta \theta_i \right\|$ for which the CP-model can *not* be bounded: although the sequence $\{ \mu_i \}$ can be decreasing, the first part of the definition of a *safe* β-strategy exactly serves to ensure that it will be finitely convergent to some $\bar{\mu} > 0$ (although $\liminf \mu_i = \bar{\mu}$ would suffice), that can be used as the ξ above.
It is now possible to show that, at length, *all the directions* $\boldsymbol{d}$ *are maximizers of the corresponding* $\varphi_\beta()$: in fact, (part of) the Complementary Slackness Conditions for $(\Pi_{\beta t})$ and $(\Delta_{\beta t})$ are

$$(\, \boldsymbol{d}\boldsymbol{G}_\beta + \alpha_\beta - v\boldsymbol{e} \,)\theta = 0$$

so that, calling $B = \{ h : \theta_h > 0 \}$ and $\Gamma_B = \{ \boldsymbol{g}_h : h \in B \}$, one has

$$\varphi_\beta(\boldsymbol{d}) = v = \boldsymbol{d}\boldsymbol{g}_h + \alpha_h \qquad \forall\, h \in B$$

i.e. *every* $\boldsymbol{g}_h$ for $h \in B$ is a 0-*subgradient of* $\varphi_\beta()$ (although not necessarily of $\varphi()$) in $\boldsymbol{d}$. Furthermore, since $\theta_h = 0$ for all $h \notin B$, $\left\| \Sigma_{i \in B}\, \boldsymbol{g}_i \theta_i \right\| = \left\| z \right\| < \bar{\mu}$ and hence Observation I.2.3.4 can be applied again to Γ_B, yielding a $\theta' \in \Theta$ such that $\boldsymbol{G}_\beta \theta' = \boldsymbol{0}$, i.e. showing that $\boldsymbol{0} \in \text{conv}\{ \Gamma_B \} \subseteq \partial\varphi_\beta(\boldsymbol{d}) \Rightarrow$ the optimal solution of $(\Pi_{\beta t})$ is a maximizer of $\varphi_\beta()$.
Now, the (finite) maximizer φ^* of $\varphi()$ is by definition the minimal among the set of all the maximizers $\bar{\varphi}^*_\beta$ of $\bar{\varphi}_\beta()$, but the *safe* β-strategy ensures that this is a *finite* set: hence, as in the above Observation, $\min\{ \bar{\varphi}^*_\beta - \varphi^* : \bar{\varphi}^*_\beta > \varphi^* \} = \bar{\eta} > 0$. But it is also true that $\{ v_i \} \to 0$, i.e. that

$$\bar{\varphi}^*_\beta - \bar{\varphi}(\bar{\boldsymbol{y}}) \geq \bar{\varphi}^*_\beta - \varphi^*$$

will become smaller than any fixed threshold: hence, we can conclude that *starting from some iteration* h*, the value* $\bar{\varphi}^*_\beta$ *predicted by the CP model will be exactly the optimal value* φ^*, or in other words that the sequence of the $\bar{\varphi}^*_\beta$ finitely converges to φ^*. Unfortunately, without other assumptions on $\varphi()$,

such as *uniqueness* of the optimal solution, this does not tell us anything about the actual value of $\varphi(\bar{\boldsymbol{y}})$: however, here is where the second part of the definition of a safe β-strategy enters to play, ensuring that after a finite number of steps a complete description of the optimal level set of $\varphi()$ is obtained, hence terminating with an optimal solution. ♦

Note that the second part of the Definition I.2.3.3 kills any possibility of working with a β of bounded dimension: eventually, in the last steps nothing better than accumulating all the subgradients can be done. Alternatively, one could require properties that (typically) guarantee uniqueness of the optimal solution, such as the *Haar constraint qualification* that is postulated in [Ki85a], but these properties are unlikely to hold (and, even in case, difficult to be proved) for practical problems.

It may be instructive to terminate this paragraph by pointing out two other possible schemes that guarantee convergence to a "true optimum" even in presence of a stopping parameter $\varepsilon > 0$: the first, directly based onto Theorem I.2.2.2, is quite obvious and it is described in Figure 13.

```
⟨ run a standard Bundle algorithm until σ + t*‖ z ‖² ≤ εφ(ȳ) ⟩; t = t*;
do
    ⟨ solve (Δ_t) and (Π_t), and let θ, (v, d) be the optimal solutions ⟩;
    if( ‖ Gθ ‖ = 0 )
      then ⟨ y* = ȳ + d is optimal ⟩; Stop;
      else t = 2·t;
while( ~ Stop );
```

Figure 13: a "dual-primal" exact convergence scheme

The "post-processing" of the "ε-optimal" solution found by the standard algorithm, i.e. the exact solution of (Δ_t), can be done by just inhibiting the SS mechanism and performing a sequence of Null Steps with t set to t^*: the t increasing mechanism accounts for the possibility of a wrong choice of t^*. Such a scheme will finitely converge to a true optimum if $\varphi()$ is a polyhedral function and a monotone β-strategy that is *also safe* (in the sense of limiting the total number of aggregations) is used: it will converge in the limit to a maximizer only if t^* has been properly chosen (the t increasing mechanism can enter into the play only after termination), but it will do it for any nondifferentiable function. Interestingly, nothing is assumed about ε, i.e. this process can be started however far from $\boldsymbol{y}^*$: this divides the Bundle algorithm into two parts, the first (*dual*) responsible for obtaining a first approximately optimal point and the second (*primal*) devoted to "refine" this approximation to the true optimum. This is pretty much the converse of a *Dantzig-Wolfe approach* using the *Phase* 0 */ Phase* 1 *mechanism*, where a bounded Cutting Plane model is constructed first by disregarding the value of $\varphi()$, and only subsequently used to optimize the actual function: this idea will be used in the next paragraph to justify some of the new t-strategies that we propose.

Finally, an entirely different approach to "exact" termination with $\varepsilon > 0$ (assuming a polyhedral $\varphi()$ and a *safe* monotone β-strategy) is just to set $m_1 = 1$, i.e. to avoid Serious Steps unless $\varphi_\beta(\boldsymbol{d}) = \varphi(\bar{\boldsymbol{y}} + \boldsymbol{d})$ $[= \phi_t(\bar{\boldsymbol{y}}) + {}^1\!/_{2t}\| \boldsymbol{d} \|^2]$: this is probably *not* a good idea in practice, but it raises the interesting and yet unanswered question of whether *dynamic adjustments* of the m_1 parameter could be useful in practice, that may deserve some research.

I.3 The (QP) Subproblem

While the previous Chapters have discussed the *theoretical* meaning of various direction finding subproblems, the present one is entirely devoted to their *efficient numerical solution*: the attention will be focused on the "standard" ($\Delta_{\beta t}$) and ($\Pi_{\beta t}$) problems of §I.1.1 and some of their generalizations. For notational convenience, a slightly different formulation will be considered throughout this Chapter: in the problems arising from *minimization* of a *convex* function, by multiplying the objective function of ($\Delta_{\beta t}$) by $1 / t$ one obtains

($\Delta_{\beta t}$) $\quad \min_{\theta} \{ \frac{1}{2} \| \boldsymbol{G}_{\beta} \theta \|^2 + (\frac{1}{t}) \alpha_{\beta} \theta : \theta \in \Theta \}$

($\Pi_{\beta t}$) $\quad \min_{d,v} \{ v + \frac{1}{2} \| \boldsymbol{d} \|^2 : v \geq \boldsymbol{dg}_i - (\frac{1}{t}) \alpha_i \;\; i \in \beta \}$

(note the changes in the signs), where each occurrence of the linearization errors is multiplied by $1 / t$. Hence, t can be "embedded" in the vector α: in this Chapter, each occurrence of α_i should be read (unless explicitly stated) as $(1/t)\alpha_i$, and the problems become

(Δ_{β}) $\quad \min_{\theta} \{ \frac{1}{2} \| \boldsymbol{G}_{\beta} \theta \|^2 + \alpha_{\beta} \theta : \theta \in \Theta \}$

(Π_{β}) $\quad \min_{d,v} \{ v + \frac{1}{2} \| \boldsymbol{d} \|^2 : v \geq \boldsymbol{dg}_i - \alpha_i \;\; i \in \beta \}$.

Both (Π_{β}) and (Δ_{β}) are Semidefinite Quadratic Problems and could be solved by standard SQP codes [GMW81] [Po83] [CLM93], but the need for low solution times calls for the development of a specialized code that exploits their valuable structural properties: this is especially true because, within a Bundle algorithm, a (long) sequence of subproblems have to be solved that only differs for the addition/deletion of a few $\boldsymbol{g}_i$s from β, the modification of α and/or of t, so that a suitable algorithm can converge in very few steps if enough care is taken in conserving information between two subsequent calls. Furthermore, in the solution of large-scale Lagrangean Duals the "long" subgradients ($m \approx 10^3$ - 10^5) are obtained by solving a (possibly hard) optimization problem, so that β is usually "small" ($p = |\beta| \approx 10^2$): hence, (Δ_{β}) has much fewer variables and simpler constraints than (Π_{β}), even though the latter has a much simpler (separable) objective function - in fact, most of the specialized algorithms proposed in the literature [Ki86] [Mo87] [Ki94] work primarily on (Δ_{β}).

In this Chapter, a new active-set method for the solution of (Δ_{β}) is proposed, that enhances upon the known specialized approaches by distinguishing among bases with different properties and exploiting their structure in order to reduce the computational cost of the basic operations: in §I.3.1 an overview of the method is given, and simple proofs of convergence are presented, while in §I.3.2 the details of

the matrix factorisation methods are discussed. In §I.3.3 the method is extended to a more general version of (Π_β) with (general) linear constraints on the dual solution $\boldsymbol{d}$, while in §I.3.4 it is shown how to reoptimize after changes in the number of variables of $\boldsymbol{d}$: this is used in §I.3.5 to develop a specialized algorithm for (Π_β) with box constraints on the dual space. Then, in §I.3.6 the relevant implementative details of the actual C++ code are discussed, and in §I.3.7 some computational comparisons are reported to show the effectiveness of the approach with respect to other non-specialized (QP) codes and the importance of an ad-hoc approach to the solution of (Δ_β) for the overall efficiency of a Bundle method.

I.3.1 An overview of the algorithm

The basic theory of the TT algorithm (whose name comes from the typical "shape" of the $\boldsymbol{G}_\beta$ matrix for this class of applications, i.e. Tall and Thin) is presented here in a simple, self-contained form: by Quadratic Duality [Ge69], a feasible solution θ of (Δ_β) is optimal if and only if

$$\boldsymbol{d} = -\boldsymbol{G}_\beta\theta \qquad \text{and} \qquad v = -\boldsymbol{d}^T\boldsymbol{d} - \alpha_\beta\theta$$

are feasible for (Π_β) and vice-versa, since for any $\theta \in \Theta$ ($\theta \geq \boldsymbol{0}$, $\boldsymbol{e}\theta = 1$) and (v, $\boldsymbol{d}$) as above such that $v\boldsymbol{e} \geq \boldsymbol{d}\boldsymbol{G}_\beta - \alpha_\beta$, the *Complementary Slackness Conditions* $(\alpha_\beta - \boldsymbol{d}\boldsymbol{G}_\beta + v\boldsymbol{e})\theta = 0$ hold, so that θ are feasible dual multipliers for (Π_β). The "primal" problem can be rewritten as

(Δ_β) $\qquad \min_\theta \{ f_\beta(\theta) = {}^1/_2\theta^T\boldsymbol{Q}_\beta\theta + \alpha_\beta\theta : \theta \in \Theta \}$

where $\boldsymbol{Q}_\beta = \boldsymbol{G}_\beta^T\boldsymbol{G}_\beta$ is a positive semidefinite matrix, i.e. the minimisation of the (non-strictly) convex function $f_\beta()$ over the unitary simplex: the algorithm is of the feasible descent type, generating a finite sequence $\{ \bar{\theta}_i \}$ of feasible points with strictly decreasing objective function values, and uses the well-known *active set* strategy [GMW81] - attaching at each step a restriction of (Δ_β) in which some of the variables are fixed to zero - to handle the constraints.

Definition I.3.1.1: *given any subset* $B \subseteq \beta$ *(a* ***base****), we set*

(Δ_B) $\qquad \min_\theta \{ f_B(\theta) = {}^1/_2\theta\boldsymbol{Q}_B\theta + \alpha_B\theta \ : \theta \in \Theta \}$

where $\theta \in \mathcal{R}^{|B|}$ *and* $\boldsymbol{Q}_B$, α_B *are appropriate submatrices of* $\boldsymbol{Q}_\beta$, α_β.

As for the bundle β, bases will be considered sets of indices as well as sets of subgradients (an element of the base will be called an *item*): some other notational simplifications, such as avoiding to distinguish sets Θ with different dimensions and omitting transpose signs, will also be made when clarity is not affected. Each step is characterised by the *base* B, the corresponding $\boldsymbol{Q}_B$, $\boldsymbol{G}_B$ ... of subproblem (Δ_B) and the *current feasible point* $\theta = [\ \theta_B\ \boldsymbol{0}\ ] \in \Theta$: at each step, a *descent direction* $\boldsymbol{w}$ (for $f_\beta()$ at θ) is found, and B is revised until a solution that is optimal for both (Δ_B) and the whole (Δ_β) is reached. The *inequality constrained* subproblem (Δ_B) is tackled by considering its *equality constrained* relaxation

(R_B) $\qquad \min_\theta \{ f_B(\theta_B) : \boldsymbol{e}\theta_B = 1 \}$

where $\bar{\theta}_B$ is an optimal solution if and only if the Kuhn-Tucker optimality conditions

$$\boldsymbol{e}\bar{\theta}_B = 1 \qquad \text{and} \qquad \exists\ \rho \in \mathcal{R} \text{ s.t. } \rho\boldsymbol{e} = \boldsymbol{Q}_B\bar{\theta}_B + \alpha_B$$

hold, i.e. iff at least a solution $[\ \bar{\theta}_B\ \rho\ ]$ of the linear system

$$\text{(KT)} \qquad \begin{bmatrix} \boldsymbol{Q}_B & -\boldsymbol{e} \\ \boldsymbol{e}^T & 0 \end{bmatrix} \begin{bmatrix} \theta_B \\ \rho \end{bmatrix} = \begin{bmatrix} -\alpha_B \\ 1 \end{bmatrix}$$

exists: finding a solution of the (KT) system is the basic step of the algorithm, described in Figure 14, and it will be shown in §I.3.2 how to (efficiently) construct either the unique $[\ \bar{\theta}_B\ \rho\ ]$, if uniqueness holds, or a direction $\boldsymbol{w}_B$ such that $\boldsymbol{w}_B\boldsymbol{Q}_B = \boldsymbol{0}$ and $\boldsymbol{e}\boldsymbol{w}_B = 0$ otherwise.

TT Algorithm

```
B = { 1 }; θ = [ 1, 0, … , 0 ];                          /* initialisation */
while( ∃ h ∉ B such that v < g_h d − α_h )                /* main iteration */
    B = B ∪ { h };                                       /* item insertion */
    do                                                   /* inner iteration */
       if( the (KT) system admits unique solution [ θ̄_B ρ ] )
           then if( θ̄_B ≥ 0 )   then θ = [ θ̄_B 0 ]; w_B = 0;
                                else w_B = θ̄_B − θ_B;
         else ⟨ find a feasible (descent) direction w_B s.t. w_B Q_B = 0 and e w_B = 0 ⟩;
       if( w_B ≠ 0 )                                     /* θ̄_B is not optimal for (Δ_B) */
           then  η = min{ −w_h / θ_h : w_h < 0 , h ∈ B  };   /* maximum feasible step η > 0 */
                 θ_B = θ_B + η w_B;                      /* take step η along w_B */
       forall( h such that θ_h = 0 )
           B = B / { h };                                /* item(s) deletion */
    while( w_B ≠ 0 );                                    /* end inner iteration */
endwhile                                                 /* end main iteration */
```

Figure 14: the TT algorithm

If (the unique) $\bar{\theta}_B$ is feasible, i.e. $\bar{\theta}_B \geq \boldsymbol{0}$, then it is optimal for (Δ_B) too, otherwise, by convexity of $f_B()$, $\boldsymbol{w}_B = \bar{\theta}_B - \theta_B$ is a direction of descent (and $\boldsymbol{e}\boldsymbol{w}_B = 0$): furthermore, it is also feasible, i.e. $\theta_B + \eta\boldsymbol{w}_B \geq \boldsymbol{0}$ for a sufficiently small $\eta > 0$. If the current inner iteration is *not* the first of a main iteration, then $\theta_B > \boldsymbol{0}$ (all the h such that $\theta_h = 0$ have been eliminated at the end of the previous inner iteration) and any $\boldsymbol{w}_B$ such that $\boldsymbol{e}\boldsymbol{w}_B = 0$ is feasible, otherwise $B = B' \cup \{\ h\ \}$, $\theta_B = [\ \theta_{B'}\ \theta_h\ ] = [\ \theta_{B'}\ 0\ ]$, $\theta_{B'} > \boldsymbol{0}$ is optimal for $(\Delta_{B'})$ and the h-th dual constraint is violated, i.e. $v < \boldsymbol{g}_h\boldsymbol{d} - \alpha_h$. In this case, it is possible to show that $\boldsymbol{w}_B = [\ \boldsymbol{w}_{B'}\ w_h\ ] = \bar{\theta}_B - [\ \theta_{B'}\ 0\ ]$ is feasible, i.e. that $w_h > 0$ (no other entry can create problems): for $\boldsymbol{d} = -\boldsymbol{G}_{B'}\theta_{B'}$ and $v = -\left\| \boldsymbol{d} \right\|^2 - \alpha_{B'}\theta_{B'}$ (the optimal solutions of $(\Pi_{B'})$), the KT conditions can be rewritten

$$-\rho\boldsymbol{e} = \boldsymbol{d}\boldsymbol{G}_{B'}^T - \alpha_{B'}$$

that, (right)multiplying both sides for $\theta_{B'}^T$ and using $\boldsymbol{e}\theta_{B'} = 1$, transforms into

$$-\rho = -\boldsymbol{d}^T\boldsymbol{d} - \alpha_{B'}\theta_{B'} = v$$

so that $\quad v = \boldsymbol{d}\boldsymbol{g}_h - \alpha_h \quad$ for each $h \in B'$

i.e. all the (dual) constraints in B' are satisfied as equations and the optimal multiplier ρ associated with the (primal) constraint $\boldsymbol{e}\theta_{B'} = 1$ is just $-v$. Then, for any $\boldsymbol{w} = [\ \boldsymbol{w}'\ w_h\ ]$ such that $\boldsymbol{e}\boldsymbol{w} = 0$

$$f(\theta_B + \eta \boldsymbol{w}) = f(\theta_{B'}) + \eta [\, \boldsymbol{w}' \; w_h \,] \left(\begin{bmatrix} \boldsymbol{Q}_{B'} & \boldsymbol{Q}_{B'h} \\ \boldsymbol{Q}_{B'h}^T & \boldsymbol{q}_{hh} \end{bmatrix} \begin{bmatrix} \theta_{B'} \\ 0 \end{bmatrix} + \begin{bmatrix} \alpha_{B'} \\ \alpha_h \end{bmatrix} \right) + \frac{1}{2} \eta^2 \boldsymbol{w}^T \boldsymbol{Q}_B \boldsymbol{w}$$

where $\boldsymbol{Q}_{B'h} = \boldsymbol{G}_B^T \boldsymbol{g}_h$: using the optimality of $\theta_{B'}$ ($\Rightarrow \boldsymbol{Q}_{B'} \theta_{B'} + \alpha_{B'} = -v\boldsymbol{e}$) and $\boldsymbol{e}\boldsymbol{w} = 1$ ($\Rightarrow \boldsymbol{e}\boldsymbol{w}' = -w_h$) the first-order term

$$\eta [\, \boldsymbol{w}' \; w_h \,] \begin{bmatrix} \boldsymbol{Q}_{B'} \theta_{B'} + \alpha_{B'} \\ \boldsymbol{g}_h^T \boldsymbol{G}_{B'} \theta_{B'} + \alpha_h \end{bmatrix}$$

can be rewritten as

$$\eta [\, w_h (\alpha_h - \boldsymbol{g}_h \boldsymbol{d}) - v(\boldsymbol{e}\boldsymbol{w}') \,] = \eta w_h [\, v + \alpha_h - \boldsymbol{g}_h \boldsymbol{d} \,]$$

that is (strictly) negative if and only if w_h is (strictly) positive, since $v + \alpha_h - \boldsymbol{g}_h \boldsymbol{d} < 0$; therefore, a direction $\boldsymbol{w}$ is feasible if and only if it is of descent, and, by convexity of $f_B()$, $\boldsymbol{w}_B$ was already know to be a direction of descent. If $\boldsymbol{w}_B \boldsymbol{Q}_B = \boldsymbol{0}$ (an *infinite direction*), from the previous formulas

$$f\left(\begin{bmatrix} \theta_{B'} \\ 0 \end{bmatrix} + \eta \begin{bmatrix} \boldsymbol{w}_{B'} \\ w_h \end{bmatrix} \right) = f(\theta_{B'}) + \eta [\, \boldsymbol{w}_{B'} \; w_h \,] \begin{bmatrix} \alpha_{B'} \\ \alpha_h \end{bmatrix}$$

and therefore one among $\pm \boldsymbol{w}_B$ (depending on the sign of the scalar product) is a direction of descent: if $\theta_B > \boldsymbol{0}$ there can be no problems, while $\theta_B = [\, \theta_{B'} \; 0 \,]$ one has

$$\eta [\, \boldsymbol{w}_{B'} \; w_h \,] \begin{bmatrix} \alpha_{B'} \\ \alpha_h \end{bmatrix} = \eta w_h [\, v + \alpha_h - \boldsymbol{g}_h \boldsymbol{d} \,]$$

and therefore by choosing the feasible one among $\pm \boldsymbol{w}_B$ (the one with $\pm w_h > 0$), a feasible direction is also obtained.

It is easy to show that the algorithm finitely terminates: only a finite number of inner iterations can be performed between two successive main iterations, since at any inner iteration (at least) one item is eliminated from the base; moreover, it is impossible for a base B to appear twice as the current base at the beginning of a main iteration, since there the current solution θ_B is optimal for (Δ_B) and at least a strictly decreasing step has been performed since the latest main iteration (a *strictly violated constraint* has been inserted in base), but the number of different bases is finite (although exponential) and this ensures finite termination. Although the basic structure of the TT algorithm is almost the same as that of [Ki86] and [Mo87], the above proof of convergence is much simpler than the ones available in the literature: actually, all these methods differ essentially for how the (KT) system is solved, and the next paragraph will show how this task is accomplished in our algorithm.

I.3.2 Solving the (KT) system

At each step, the (KT) system is solved (or an infinite direction is determined) by essentially keeping a *lower trapezoidal factorisation* of the Hessian matrix $\boldsymbol{Q}_B$ of the current subproblem (Δ_B), that is

$$\bar{\boldsymbol{L}}_B = \begin{bmatrix} \boldsymbol{L}_{B'} & \boldsymbol{0} \\ \boldsymbol{V}^T & 0 \end{bmatrix} \qquad \text{such that} \qquad \bar{\boldsymbol{L}}_B \bar{\boldsymbol{L}}_B^T = \boldsymbol{Q}_B$$

where $\boldsymbol{L}_{B'}$ is a lower triangular matrix with all nonzero diagonal entries: actually, $\boldsymbol{V}$ always has only

up to two columns, i.e. $\bar{\boldsymbol{L}}_B$ has only the following three possible configurations

(0) $B = B'$	(1) $B = B' \cup \{ h \}$	(2) $B = B' \cup \{ h \} \cup \{ j \}$
$\bar{\boldsymbol{L}}_B = \boldsymbol{L}_{B'}$	$\bar{\boldsymbol{L}}_B = \begin{bmatrix} \boldsymbol{L}_{B'} & \boldsymbol{0} \\ \boldsymbol{l}_h^T & 0 \end{bmatrix}$	$\bar{\boldsymbol{L}}_B = \begin{bmatrix} \boldsymbol{L}_{B'} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{l}_h^T & 0 & 0 \\ \boldsymbol{l}_j^T & 0 & 0 \end{bmatrix}$

The rationale is that B' is the subset of B containing *linearly independent items* $\boldsymbol{g}_i$, while $\boldsymbol{g}_h$ (and $\boldsymbol{g}_j$) are linearly dependent from the items in B': in fact, the (KT) system may have full row rank (hence unique optimal solution) even if $\boldsymbol{Q}_B$ is 1-rank deficitary, while if B contains two linearly dependent items then the (KT) system is underdetermined and a feasible infinite direction $\boldsymbol{w}_B$ can be found. The basic relations are summarised in the following propositions, whose proofs are just linear algebra, and can be omitted.

Proposition I.3.2.1: *in case (0), the unique solution of (KT) is*

$$\bar{\theta}_B = \boldsymbol{Q}_B^{-1}[\, \rho \boldsymbol{e} - \alpha_B \,] \qquad \textit{where} \qquad \rho = \frac{1 + \boldsymbol{e}\, \boldsymbol{Q}_B^{-1} \alpha_B}{\boldsymbol{e}\, \boldsymbol{Q}_B^{-1} \boldsymbol{e}}$$

Note that, defining

$$\boldsymbol{z}_1 = \boldsymbol{L}_{B'}^{-1} \boldsymbol{e} \qquad \text{and} \qquad \boldsymbol{z}_2 = \boldsymbol{L}_{B'}^{-1} \alpha_B$$

(the *unique* solutions of the triangular systems $\boldsymbol{L}_B \boldsymbol{z}_1 = \boldsymbol{e}$, $\boldsymbol{L}_B \boldsymbol{z}_2 = \alpha_{B'}$), one has

$$\bar{\theta}_B = (\boldsymbol{L}_B^T)^{-1}[\, \rho \boldsymbol{z}_1 - \boldsymbol{z}_2 \,] \qquad \text{where} \qquad \rho = \frac{1 + \boldsymbol{z}_1 \boldsymbol{z}_2}{\boldsymbol{z}_1 \boldsymbol{z}_1}$$

i.e. $\bar{\theta}_B$ can be obtained at the cost of three backsolves on the triangular matrix $\boldsymbol{L}_{B'}$ and a few linear operations: actually, by exploiting the base of the previous step, this will be reduced to only one backsolve.

Proposition I.3.2.2: *in case (1), if* $\boldsymbol{l}_h \boldsymbol{z}_1 \neq 1$ *then the unique solution of (KT) is*

$$\bar{\theta}_h = \frac{1 + \boldsymbol{z}_1 \boldsymbol{z}_2 - \rho \boldsymbol{z}_1 \boldsymbol{z}_1}{1 - \boldsymbol{l}_h \boldsymbol{z}_1}, \; \bar{\theta}_{B'} = (\boldsymbol{L}_{B'}^T)^{-1}[\, \rho \boldsymbol{z}_1 - \boldsymbol{z}_2 - \bar{\boldsymbol{x}}_h \boldsymbol{l}_h \,] \qquad \textit{where } \rho = \frac{\alpha_h - \boldsymbol{l}_h \boldsymbol{z}_2}{1 - \boldsymbol{l}_h \boldsymbol{z}_1}.$$

Proposition I.3.2.3: *in case (1), if* $\boldsymbol{l}_h \boldsymbol{z}_1 = 1$ *then an infinite direction is*

$$\boldsymbol{w}_B = [\, \boldsymbol{w}_{B'} \;\; w_h \,] = [\, \boldsymbol{v}_1 \;\; -1 \,] \qquad \textit{where } \boldsymbol{v}_1 = (\boldsymbol{L}_{B'}^T)^{-1} \boldsymbol{l}_h.$$

Proposition I.3.2.4: *in case (2), if* $\boldsymbol{l}_j \boldsymbol{z}_1 = 1$ *then an infinite direction is*

$$\boldsymbol{w}_B = [\, \boldsymbol{w}_{B'} \;\; w_h \;\; w_j \,] = [\, \boldsymbol{v}_2 \;\; 0 \;\; -1 \,] \;\; \textit{where } \boldsymbol{v}_2 = (\boldsymbol{L}_{B'}^T)^{-1} \boldsymbol{l}_j.$$

Proposition I.3.2.5: *in case (2), if* $\boldsymbol{l}_j z_1 \neq 1$ *then an infinite direction is*

$$\boldsymbol{w}_B = [\ \boldsymbol{v}\ \ -1\ \ \mu\ ] \qquad \textit{where} \quad \mu = \frac{1 - \boldsymbol{l}_h z_1}{1 - \boldsymbol{l}_j z_1} \ \textit{and} \qquad \boldsymbol{v} = \boldsymbol{v}_1 - \mu \boldsymbol{v}_2 = (\boldsymbol{L}_{B'}^T)^{-1}[\ \boldsymbol{l}_h - \mu \boldsymbol{l}_j\ ].$$

Note that in case (2) an infinite direction $\boldsymbol{w}_B$ can always be found: this immediately implies that B will never contain more than two linearly dependent items, since when the second one is inserted in B, a sequence of inner iterations (using infinite directions) starts, and at each iteration at least one item is deleted from B until full rank of the (KT) system is restored. Actually, it may happen that some of the infinite directions used by the algorithm be not *strictly descent* (if $\alpha_B \boldsymbol{w}_B = 0$), but this cannot happen in the first inner iteration of a main iteration since the selected h-th constraint is *strictly* violated, and hence convergence is not affected. Since at the beginning of each main iteration

$$\boldsymbol{Q}_B \bar{\theta}_B = \rho \boldsymbol{e} - \alpha_B \quad \Rightarrow \quad \bar{\theta}_B \boldsymbol{Q}_B \bar{\theta}_B = \rho - \bar{\boldsymbol{x}}_B \alpha_B \ \Rightarrow\ f_B(\ \bar{\theta}_B\ ) = {}^1\!/_2 \bar{\theta}_B \boldsymbol{Q}_B \bar{\theta}_B + \alpha_B \bar{\theta}_B = {}^1\!/_2(\ \rho - \bar{\theta}_B \alpha_B\ )$$

the cost of the $f_B(\ \bar{\theta}_B\ )$ calculation can be reduced from $O(\ b^2\ )$ to $O(\ b\ )$ ($b = |\ B\ |$): this could be done at any inner iteration also, since $f_B()$ is linear when restricted along an infinite $\boldsymbol{w}_B$ (as shown in §I.3.1), and if $\boldsymbol{w}_B = \bar{\theta}_B - \theta_B$

$$f_B(\ \theta_B + \eta \boldsymbol{w}_B\ ) = f_B(\ \theta_B\ ) + \eta w_h[\ v + \alpha_h - \boldsymbol{g}_h \boldsymbol{d}\ ] + {}^1\!/_2 \eta^2 [\ f_B(\ \theta_B\ ) - f_B(\ \bar{\theta}_B\ )\ ]$$

but this is not useful (and probably error-prone), and it is avoided in the current implementation.

Once the lower trapezoidal matrix $\bar{\boldsymbol{L}}_B$ and the vectors z_1, z_2 have been found, they can be keet updated at low cost whenever an item is inserted into, or deleted from, B: therefore, the above propositions show that $\bar{\theta}_B$ or $\boldsymbol{w}_B$ can be obtained at each step at the cost of a single backsolve on $\boldsymbol{L}_{B'}$ (at the first iteration $\bar{\boldsymbol{L}}_B = \boldsymbol{L}_{B'} = [\ 1\ /\ \boldsymbol{Q}_{11}\ ]$ and z_1, z_2 can be obtained accordingly). The method is described by the following propositions:

Proposition I.3.2.6: *for* $B = B' \cup \{\ h\ \}$ *and* $\bar{L}_B = L_B$ *nonsingular*

$$\boldsymbol{L}_B = \begin{bmatrix} \boldsymbol{L}_{B'} & \boldsymbol{0} \\ \boldsymbol{l}_h^T & \delta \end{bmatrix} \qquad \textit{where} \quad \boldsymbol{l}_h = (\boldsymbol{L}_{B'}^{-1})[\ \boldsymbol{G}_B^T \boldsymbol{g}_h\ ] \qquad \textit{and} \qquad \delta = \sqrt{\|\ \boldsymbol{g}_h\ \|^2 - \|\ \boldsymbol{l}_h\ \|^2};$$

hence, $\boldsymbol{L}_B$ *is nonsingular* $\Leftrightarrow \delta > 0 \ \Leftrightarrow \boldsymbol{g}_h$ *is linearly independent from the items in* B'.

This follows from the rules of Cholesky factorisation applied to $\boldsymbol{Q}_B = \boldsymbol{G}_B^T \boldsymbol{G}_B$: for a sketch, note that

$$\|\ \boldsymbol{l}_h\ \|^2 = (\boldsymbol{g}_h^T \boldsymbol{G}_B \boldsymbol{L}_{B'}^{-1T})(\boldsymbol{L}_{B'}^{-1} \boldsymbol{G}_B^T \boldsymbol{g}_h) = \boldsymbol{g}_h^T [\ \boldsymbol{G}_{B'} (\boldsymbol{G}_{B'}^T \boldsymbol{G}_{B'})^{-1} \boldsymbol{G}_{B'}^T\ ] \boldsymbol{g}_h = [\ \boldsymbol{g}_h^T \boldsymbol{K}_{B'}\ ] \boldsymbol{g}_h = \bar{\boldsymbol{g}}^T \boldsymbol{g}_h$$

where $\boldsymbol{K}_{B'}$ is the projection operator onto the subspace $\Gamma_{B'}$ spanned by the vectors in B'. Hence, $\bar{\boldsymbol{g}}$ is the projection of $\boldsymbol{g}_h$ onto $\Gamma_{B'}$, so that $\|\ \boldsymbol{l}_h\ \|^2 = \bar{\boldsymbol{g}}^T \boldsymbol{g}_h \le \|\ \boldsymbol{g}_h\ \|^2$ and equality holds $\Leftrightarrow \boldsymbol{g}_h \in \Gamma_{B'}$ i.e. $\boldsymbol{g}_h$ is linearly dependent from the items in B'. ♦

This is exactly one step of the *row-wise* LL^T factorization algorithm applied to $\boldsymbol{Q}_\beta$: to extend the result to the case where $\bar{\boldsymbol{L}}_B$ is lower trapezoidal, consider a p-vector γ and its subvector $\gamma_{B'}$.

Proposition I.3.2.7: *if* $\boldsymbol{L}_B \boldsymbol{r}_{B'} = \gamma_{B'}$, *then*

$$\boldsymbol{L}_B \begin{bmatrix} \boldsymbol{r}_{B'} \\ \dfrac{\gamma_h - \boldsymbol{l}_h \boldsymbol{r}_{B'}}{\delta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}_{B'} & \boldsymbol{0} \\ \boldsymbol{l}_h & \delta \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_{B'} \\ \dfrac{\gamma_h - \boldsymbol{l}_h \boldsymbol{r}_{B'}}{\delta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}_B \boldsymbol{r}_{B'} \\ \boldsymbol{l}_h \boldsymbol{r}_{B'} + \delta \left(\dfrac{\gamma_h - \boldsymbol{l}_h \boldsymbol{r}_{B'}}{\delta} \right) \end{bmatrix} = \begin{bmatrix} \gamma_{B'} \\ \gamma_h \end{bmatrix} = \gamma_B .$$

In other words, $\boldsymbol{r}_B = \boldsymbol{L}_B^{-1} \gamma_B = [\, \boldsymbol{r}_{B'} \;\; r_h \,]$ where $\boldsymbol{r}_{B'}$ is known and r_h can be calculated in O($b' = \left| B' \right|$): the importance of this proposition lies in the fact that all the "relevant" vectors $\boldsymbol{l}_h$, $\boldsymbol{l}_j$, $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ have the above form (for γ respectively equal to $\boldsymbol{Q}_h$, $\boldsymbol{Q}_j$, α and $\boldsymbol{e}$), and therefore can be cheaply updated when inserting a new item in B. Analogously, the scalar products among such vectors that are used within the algorithm (such as $\boldsymbol{l}_h \boldsymbol{z}_1$, $\boldsymbol{z}_1 \boldsymbol{z}_2$ ...) can also be updated in constant time when the new entries are calculated, saving some O(b) computations.

The above relations also shed some light on Propositions I.3.2.2 - I.3.2.5: for example, the "critical" scalar product $\boldsymbol{l}_h \boldsymbol{z}_1$ of Proposition I.3.2.2 can be rewritten as $\boldsymbol{l}_h^T [\, \boldsymbol{L}_{B'}^{-1} \boldsymbol{e} \,] = \boldsymbol{e} [\, \boldsymbol{L}_{B'}^{-1} \boldsymbol{l}_h \,] = \boldsymbol{e} \boldsymbol{v}_1$, so that the condition $\boldsymbol{l}_h \boldsymbol{z}_1 = 1$ is equivalent to $[\, \boldsymbol{v}_1 \; -1 \,]^T [\, \boldsymbol{e} \;\; 1 \,] = 0$, i.e. to the feasibility of the (infinite) direction $[\, \boldsymbol{v}_1 \; -1 \,]$ w.r.t. the constraint $\boldsymbol{e} \theta_B = 1$; similar interpretations can be derived for all the other cases.

Obviously, a cheap updating procedure is also needed for the case $B = B' / \{ h \}$, where

$$\boldsymbol{L}_{B'} = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{v}^T & \delta & \boldsymbol{0} \\ \boldsymbol{Z} & \boldsymbol{w} & \boldsymbol{L}_2 \end{bmatrix}, \quad \boldsymbol{G}_{B'}^T = \begin{bmatrix} \boldsymbol{G}_1^T \\ \boldsymbol{g}_h^T \\ \boldsymbol{G}_2^T \end{bmatrix}, \quad \boldsymbol{Q}_{B'} = \boldsymbol{G}_{B'}^T \boldsymbol{G}_{B'} = \begin{bmatrix} \boldsymbol{G}_1^T \boldsymbol{G}_1 & \boldsymbol{G}_1^T \boldsymbol{g}_h & \boldsymbol{G}_1^T \boldsymbol{G}_2 \\ \boldsymbol{g}_h^T \boldsymbol{G}_1 & \| \boldsymbol{g}_h \|^2 & \boldsymbol{g}_h^T \boldsymbol{G}_2 \\ \boldsymbol{G}_2^T \boldsymbol{G}_1 & \boldsymbol{G}_2^T \boldsymbol{g}_h & \boldsymbol{G}_2^T \boldsymbol{G}_2 \end{bmatrix}$$

and $\boldsymbol{L}_B = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} \\ \boldsymbol{Z} & \boldsymbol{L}_2' \end{bmatrix}$, $\boldsymbol{G}_B^T = \begin{bmatrix} \boldsymbol{G}_1^T \\ \boldsymbol{G}_2^T \end{bmatrix}$, $\boldsymbol{Q}_B = \begin{bmatrix} \boldsymbol{G}_1^T \boldsymbol{G}_1 & \boldsymbol{G}_1^T \boldsymbol{G}_2 \\ \boldsymbol{G}_2^T \boldsymbol{G}_1 & \boldsymbol{G}_2^T \boldsymbol{G}_2 \end{bmatrix}$;

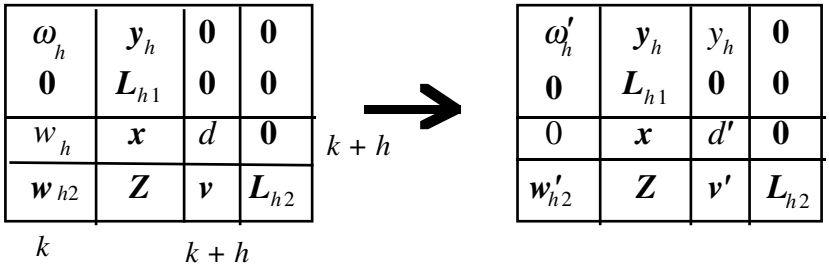
since the first $k - 1$ (where k is the position of $\boldsymbol{g}_h$ in $\boldsymbol{G}_{B'}$) rows and columns of $\boldsymbol{L}_{B'}$ are not altered by the operation, only $\boldsymbol{L}_2$ changes as $\boldsymbol{L}_2' \boldsymbol{L}_2'^T = \boldsymbol{L}_2 \boldsymbol{L}_2^T + \boldsymbol{w} \boldsymbol{w}^T$, so that it can be retrieved by a *rank-one correction* of the *triangular matrix* $\boldsymbol{L}_2'$ performed with a sweep of *Givens Rotations*.

Proposition I.3.2.8: *there exists a* $b' - k$ *orthogonal square matrix* $\boldsymbol{G}_S$ (*i.e.* $\boldsymbol{G}_S^{-1} = \boldsymbol{G}_S^T$) *such that*

$$\begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{v}^T & \delta & \boldsymbol{0} \\ \boldsymbol{Z} & \boldsymbol{w} & \boldsymbol{L}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{k-1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & & \\ \boldsymbol{0} & & \boldsymbol{G}_S \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{v}^T & \beta & \boldsymbol{y}^T \\ \boldsymbol{Z} & \boldsymbol{0} & \boldsymbol{L}_2' \end{bmatrix}$$

which implies $\boldsymbol{L}_2' \boldsymbol{L}_2'^T = \boldsymbol{L}_2 \boldsymbol{L}_2^T + \boldsymbol{w} \boldsymbol{w}^T$.

Proof: $\boldsymbol{G}_S$ is the product of $b' - k$ Givens matrices: the h-th matrix performs the transformation



so that at the end all the entries of the k-th column of $\boldsymbol{L}_{B'}$ (but $\boldsymbol{L}_{kk}$) are zero, and eliminating the k-th

row and column we obtain $\boldsymbol{L}_B$. ♦

As the h-th step requires $4(\, b - k - h \,)$ multiplications, the overall cost is $O(\, 2(\, b - k \,)^2 \,)$; furthermore, the same method can also be used to update the vectors $\boldsymbol{l}_h$, $\boldsymbol{l}_j$, z_1 and z_2:

Proposition I.3.2.9: *for $\boldsymbol{G}_S$ as in Proposition I.3.2.8, $\boldsymbol{r}_{B'}$ such that*

$$\boldsymbol{L}_B \boldsymbol{r}_{B'} = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{v}^T & \delta & \boldsymbol{0} \\ \boldsymbol{Z} & \boldsymbol{w} & \boldsymbol{L}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ r_h \\ \boldsymbol{r}_2 \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \gamma_h \\ \gamma_2 \end{bmatrix} = \gamma_B \; and \qquad \begin{bmatrix} \boldsymbol{r}_1 \\ r'_h \\ \boldsymbol{r}'_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{k-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}_S^T \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ r_h \\ \boldsymbol{r}_2 \end{bmatrix}$$

the vector $\boldsymbol{r}_B = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}'_2 \end{bmatrix}$ *solves* $\boldsymbol{L}_B \boldsymbol{r}_B = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} \\ \boldsymbol{Z} & \boldsymbol{L}'_2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta'_2 \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix} = \gamma_B$.

In other words, by applying the same Givens rotations to the old $\boldsymbol{r}_{B'}$, the new last $b' - k$ entries of $\boldsymbol{r}_B$ are obtained; it is also easy to show that, for instance,

$$\bar{z}_1 \bar{z}_2 = z_1 z_2 - z'_{1h} z'_{2h}$$

therefore $z_1 z_2$ can be kept updated in $O(\, 1 \,)$ in this case also - analogous results hold for any scalar product of this kind. Some other details have to be taken into account: for instance, in case (1) the elimination of an item from the base may make $\boldsymbol{g}_h$ to "enter the linearly dependent part" of B, i.e.

$$\begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{v}^T & \beta & \boldsymbol{y}^T & 0 \\ \boldsymbol{Z} & \boldsymbol{0} & \boldsymbol{L}'_2 & \boldsymbol{0} \\ \boldsymbol{l}_1 & l'_h & \boldsymbol{l}'_2 & 0 \end{bmatrix} \qquad \Rightarrow \qquad \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{Z} & \boldsymbol{L}'_2 & \boldsymbol{0} \\ \boldsymbol{l}_1 & \boldsymbol{l}'_2 & l'_h \end{bmatrix}$$

but this is faced by a simple $l'_h \neq 0$ and the few logical operations required to adjust the indices sets; similarly, in case (2) $\boldsymbol{g}_j$ may become independent, or $\boldsymbol{g}_h$ may be deleted from B.

Reoptimization is natural within the TT algorithm: after that (Δ_β) has been solved, the optimal base B, the optimal solution $[\, \theta_B \; \boldsymbol{0} \,]$, the lower trapezoidal factorisation $\bar{\boldsymbol{L}}_B$ of $\boldsymbol{Q}_B$ and the vectors z_1, z_2 (with the relative scalar products) are kept, and they are updated as modifications occurs to the data of the problem. In our implementation the calling program is allowed to

- add a single new item $\boldsymbol{g}_h$ (a newly obtained subgradient) to β: hence, by skipping the initialisation phase the algorithm is restarted with the optimality test phase, in which all the data structures are used "as is" to test the violation of the corresponding dual constraint;
- delete a single item $\boldsymbol{g}_h$ (an outdated subgradient) from β: if $h \notin B$ this has no effect on the problem, since θ_B obviously remains optimal, while if $h \in B$ the procedures described in Propositions I.3.2.8 - I.3.2.9 (actually, the same piece of code) are used to update all the data structures in such a way that the algorithm can be restarted at the beginning of an inner iteration, just as if $\boldsymbol{g}_h$ had been deleted from B in the course of a normal iteration of the algorithm;
- change the vector α (i.e. the current point $\bar{\boldsymbol{y}}$): then, the vector z_2 can be recalculated with just one backsolve on the linearly independent matrix $\boldsymbol{L}_{B'}$, and again the algorithm can be simply restarted at

the beginning of an inner iteration, since all the other data structures are still coherent.

Any sequence of such operations leaves the data structures in a *consistent state*: in case of a warm start, the (data structures of the) previous optimal solution can be fully exploited. Some minor details may also be noticeable: if the parameter t is explicitly handled, as in our implementation, changing t does not force the recalculation of z_2, if $z = \boldsymbol{L}_B^{-1}\alpha$ is kept and $z_2 = (1 / t)z$ is calculated only when required; furthermore, if α is changed "along" the latest optimal solution $\boldsymbol{d}$, i.e. $\boldsymbol{y} = \bar{\boldsymbol{y}} + \tau\boldsymbol{d}$, then the new vector is obtained according to the formula

$$\alpha' = \alpha - \tau(\boldsymbol{d}\boldsymbol{G}^T) + \Delta_o \boldsymbol{e}$$

where $\Delta_o = \varphi(\bar{\boldsymbol{y}} + \tau\boldsymbol{d}) - \varphi(\bar{\boldsymbol{y}})$ is the *obtained increase* in the nondifferentiable function for a *step of* τ along $\boldsymbol{d}$, and

$$\alpha'_B = \alpha_B + \tau(\boldsymbol{Q}_B\theta_B) + \Delta_o\boldsymbol{e} = \alpha_B + \tau(\rho\boldsymbol{e} - (1/t)\alpha_B) + \Delta_o\boldsymbol{e} =$$

$$= \left(1 - \frac{\tau}{t}\right)\alpha_B + (\Delta_o - \tau v)\boldsymbol{e} = \left(1 - \frac{\tau}{t}\right)\alpha_B + \left(\Delta_o - \frac{\tau}{t}\Delta_e\right)\boldsymbol{e}$$

(where $\Delta_e = tv = -t\rho$ is the *expected increase* for a step of t along $\boldsymbol{d}$), so that $z'_2 = \boldsymbol{L}_B^{-1}\alpha'$ can be obtained in linear time from z_1 and z_2. This ends the description of the "basic" TT algorithm for the unconstrained minimization case: several extensions to more general cases are possible, as the next three paragraphs show.

I.3.3 Handling general dual constraints

As discussed in more details in §I.4.3, Bundle algorithms can be extended to the *constrained* minimisation of $\varphi()$ over a polyhedron $Y = \{ \boldsymbol{y} : \boldsymbol{h} \geq \boldsymbol{y}\boldsymbol{H} \} \subseteq \mathcal{R}^m$, provided that the constraints are inserted into the (Π_β) subproblem: therefore, the (QP) solver must be capable of efficiently deal with general linear constraints on the "dual" space. As for the subgradients, only a subset γ of the constraint set (i.e. a subsystem $\boldsymbol{h}_\gamma \geq \boldsymbol{y}\boldsymbol{H}_\gamma$) may be known at any given point of the algorithm, either because not all the constraints are known a priori, or because some form of constraint generation is used to keep their number small: hence, the problem to be solved is

$$(\Pi_{\beta\gamma}) \qquad \min_{d,v} \{ v + 1/2 \| \boldsymbol{d} \|^2 : v\boldsymbol{e} \geq \boldsymbol{d}\boldsymbol{G}_\beta - \alpha_\beta \ , \boldsymbol{h}_\gamma \geq (\bar{\boldsymbol{y}} + \boldsymbol{d})\boldsymbol{H}_\gamma \}$$

or, equivalently, its dual

$$(\Pi_{\beta\gamma}) \qquad \min_\theta \{ 1/2 \| \boldsymbol{G}_\beta\theta + \boldsymbol{H}_\gamma\mu \|^2 + \alpha_\beta\theta + \bar{\boldsymbol{h}}_\gamma\mu : \theta \in \Theta \ , \mu \geq \boldsymbol{0} \}$$

where $\bar{\boldsymbol{h}}_\gamma = \bar{\boldsymbol{y}}\boldsymbol{H}_\gamma - \boldsymbol{h}_\gamma$, i.e. the inequalities are properly "translated" w.r.t. the current point to ensure feasibility of all the points $\bar{\boldsymbol{y}} + \tau\boldsymbol{d}$ with $\tau \leq t$.
Let $\boldsymbol{i}$ be the characteristic vector of the set (of "true" subgradients) β, i.e.

$$i_h = \begin{cases} 1 & \text{if } h \in \beta \\ 0 & \text{if } h \in \gamma \end{cases} :$$

the feasible set of $(\Delta_{\beta\gamma})$ can be written

$$\{ [\theta \ \mu] : \boldsymbol{i}[\theta \ \mu] = 1 \ , [\theta \ \mu] \geq \boldsymbol{0} \}$$

and *the TT algorithm can still be applied*, provided that *the all-one vector* $\boldsymbol{e}$ *is replaced in every expression with* $\boldsymbol{i}_B$ (where $B \subseteq \beta \cup \gamma$). In fact, $\boldsymbol{Q}_B\bar{\theta}_B + \alpha_B = \rho\boldsymbol{i}_B$ and $\boldsymbol{i}_B\bar{\theta}_B = 1$ are the (KT) conditions for (Δ_B), so that, letting $z_1 = \boldsymbol{L}_{B'}^{-1}\boldsymbol{i}_{B'}$ the optimal solution to (R_B) in case (0) is

$$\bar{\theta}_B = \boldsymbol{Q}_B^{-1}[\ \rho\boldsymbol{i}_B - \alpha_B\ ] \qquad \text{where still} \qquad \rho = \frac{1 + z_1 z_2}{z_1 z_1};$$

Propositions I.3.2.2 - I.3.2.3 hold true if the condition $\boldsymbol{l}_h z_1 \neq 1$ is replaced by $\boldsymbol{l}_h z_1 \neq \boldsymbol{i}_h$ and the infinite direction $[\ \boldsymbol{v}_1\ -1\ ]$ is replaced by $[\ \boldsymbol{v}_1\ -\boldsymbol{i}_h\ ]$, and an analogous treatment can be deserved to Propositions I.3.2.4 - I.3.2.5. Moreover, the factorisation methods described in §I.3.2 are obviously still valid for the constrained case, since z_1 (that solves $\boldsymbol{L}_B z_1 = \boldsymbol{i}_{B'}$) has the form required by Propositions I.3.2.7 and I.3.2.9. Minor modifications to the algorithm are needed, since it is necessary that the base B always contains at least one subgradient $\boldsymbol{g}_i$ ($i \in \beta$), but it is easy to see that once this is ensured for the initial base, it will automatically hold true thereafter; every detail of the algorithm naturally extends to the new case, for instance movements "along" $\boldsymbol{d}$ as introduced at the end of the §I.3.2 now have the α updating formula

$$\alpha' = \alpha - \tau(\ \boldsymbol{d}\boldsymbol{G}^T\ ) + \Delta_o\boldsymbol{i}$$

so that z_2' is obtained in the same way from z_1 and z_2, changes of the set γ between two subsequent calls to the ($\Delta_{\beta\gamma}$)-solver are completely analogous to changes of the set β (up to the point that the two things can be handled with the same piece of code) and so on.

These results can be further generalised by recognising that ($\Delta_{\beta\gamma}$) is just a special case of

$$\min_\theta \left\{\ {}^1/_2 \| \boldsymbol{G}\theta \|^2 + \alpha\theta : \boldsymbol{c}\theta = 1\ ,\ \theta \geq \boldsymbol{0}\ \right\}$$

where $\boldsymbol{c}$ is any vector of nonnegative coefficients: it is easy to show that all the above theory is valid if $\boldsymbol{e}$ (or $\boldsymbol{i}$) is replaced everywhere with $\boldsymbol{c}$ in the formulas. This is interesting since Least-Square-like problems frequently appear in various fields (see e.g. [FV94] for an application to fractal compression of images), and the TT algorithm can be easily extended to this case: the changes to be made are so few and so localised that this possibility is already provided by our implementation, via a "compile time switch" (see §I.3.6).

I.3.4 Changing the dimension of the dual space

The TT algorithm can work without accessing the vectors $\boldsymbol{g}_i$, since all the computations can be (efficiently) performed by means of the scalar products $\boldsymbol{q}_{ij} = \boldsymbol{g}_i^T\boldsymbol{g}_j$: by $\boldsymbol{d} = -\boldsymbol{G}_B\theta_B$, one obtains

$$\boldsymbol{d}\boldsymbol{g}_h = -[\ \boldsymbol{g}_h^T\boldsymbol{G}_B\ ]\theta_B = -\boldsymbol{Q}_{Bh}\theta_B$$

that can be used for checking the dual constraints $\boldsymbol{d}\boldsymbol{g}_h - v \leq \alpha_h$ without explicitly forming $\boldsymbol{d}$ (if $b < m$ this is also computationally convenient, since it reduces the complexity of the feasibility check from O(bm) to O(b^2)). Yet, when the algorithm terminates the scalar product $\boldsymbol{d}\boldsymbol{g}_h$ is known for each $h \in \beta$ (it has been computed during the feasibility check if $h \notin B$, and $\boldsymbol{d}\boldsymbol{g}_h = \rho\boldsymbol{i}_h - \alpha_h$ if $h \in B$), so that it can be used, for instance, for changing α_h "along" $\boldsymbol{d}$; furthermore, the NDO algorithm should have granted access to it, since $\varphi_{\beta,\boldsymbol{d}}(\tau) = \min_{h \in \beta}\{\ \tau(\boldsymbol{d}\boldsymbol{g}_h) + \alpha_h\ \}$ is the restriction of $\varphi_\beta()$ along the (tentative descent) direction $\boldsymbol{d}$.

Since the TT algorithm has in fact no information on the "real" vectors $\boldsymbol{g}_h$, it is possible to design its implementation (see §I.3.6) to be "orthogonal" to thire structure by simply demanding the task to a user-provided subroutine that computes $\boldsymbol{q}_{ij}$: furthermore, it is also possible to change their length between two subsequent calls to the TT-solver, i.e. to add or remove entries from all the $\boldsymbol{g}_h$ (and $\boldsymbol{h}_h$) at the same time. This is really useful, both for the extensions that will be discussed in §I.3.5 and for the fact that NDO solvers may benefit from *active set* strategies, where only an (hopefully small) subset of the variables $\boldsymbol{y}$ - that is dynamically revised - is considered at any given step: as shown in §I.3.7, this approach can be crucial for the effectiveness of a NDO code.

Insertions and deletions of entries in (all) the subgradients $\boldsymbol{g}_h$ (treatment of constraints $\boldsymbol{h}_h$ is identical) can be "naturally" performed with cheap data structures updates, analogous to the ones described §I.3.2, provided that a - not necessarily optimal, if other changes have already occurred - base B and the related $[\ \theta_B\ \mathbf{0}\ ]$, $\boldsymbol{Q}_\beta$, $\bar{\boldsymbol{L}}_B$, z_1, z_2 ... are kept in a consistent state. When adding a new entry, the new Hessian matrix $\tilde{\boldsymbol{Q}}_\beta = \boldsymbol{Q}_\beta + \tilde{\boldsymbol{g}}\ \tilde{\boldsymbol{g}}^T$ (where $\tilde{g}_h = \langle$new entry of $\boldsymbol{g}_h\ \rangle$) can be obtained in O(p^2), and the update of $\bar{\boldsymbol{L}}_B$, z_1, z_2 can be performed with a sweep of p Givens rotations ($\boldsymbol{G}_S$), as

$$\tilde{\boldsymbol{Q}}_B = \boldsymbol{Q}_B + \tilde{\boldsymbol{g}}_B \tilde{\boldsymbol{g}}_B^T = [\ \tilde{\boldsymbol{g}}_B\ \bar{\boldsymbol{L}}_B\ ] \begin{bmatrix} \tilde{\boldsymbol{g}}_B^T \\ \bar{\boldsymbol{L}}_B^T \end{bmatrix}$$

so that the *rank-one correction* of the lower trapezoidal matrix $\bar{\boldsymbol{L}}_B$

$$\begin{bmatrix} \tilde{\boldsymbol{g}}_{B'}\ \boldsymbol{L}_{B'} \\ \tilde{g}_h\ \ \boldsymbol{l}_h^T \\ \tilde{g}_j\ \ \boldsymbol{l}_j^T \\ 0\ \ z_1 \\ 0\ \ z_2 \end{bmatrix} \boldsymbol{G}_S \qquad \Rightarrow \qquad \begin{bmatrix} \mathbf{0}\ \ \tilde{\boldsymbol{L}}_{B'} \\ \delta_h\ \tilde{\boldsymbol{l}}_h^T \\ \delta_j\ \tilde{\boldsymbol{l}}_j^T \\ \sigma_1\ \tilde{z}_1 \\ \sigma_2\ \tilde{z}_2 \end{bmatrix}$$

suffices; note that the *position* of the *new entry* plays absolutely *no role* in the formulas, making it possible to insert (and delete) entries in any order whatsoever. Once $\tilde{\boldsymbol{g}}_{B'}$ has been nullified, again an O(p^2) task, the new $\tilde{\boldsymbol{L}}_{B'}$ and the related $\tilde{\boldsymbol{l}}_h$, $\tilde{\boldsymbol{l}}_j$, $\tilde{z}_1$, $\tilde{z}_2$ can be made "consistent" - if they are not already - with the data at hand: if $\delta_h > 0$, then the corresponding item must be inserted into the "linearly independent part" of B ($\tilde{B}' = B' \cup \{\ h\ \}$), i.e.

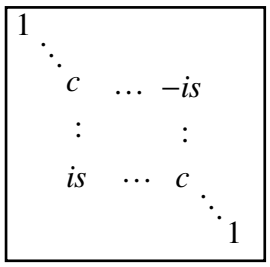
$$\boldsymbol{L}_{\tilde{B}'} = \begin{bmatrix} \tilde{\boldsymbol{L}}_{B'}\ \ \mathbf{0} \\ \tilde{\boldsymbol{l}}_h^T\ \delta_h \end{bmatrix}$$

where σ_1, σ_2 are the new entries of $\tilde{z}_1$, $\tilde{z}_2$ (so that also $\tilde{z}_1 \tilde{z}_2 = z_1 z_2 + \sigma_1 \sigma_2$); the case $\delta_j > 0$ is analogous.

When an entry (contained in $\tilde{\boldsymbol{g}}$) has to be deleted, one has $\tilde{\boldsymbol{Q}}_\beta = \boldsymbol{Q}_\beta - \tilde{\boldsymbol{g}}\ \tilde{\boldsymbol{g}}^T$, i.e.

$$\tilde{\boldsymbol{Q}}_B = [\ i\tilde{\boldsymbol{g}}_B\ \bar{\boldsymbol{L}}_B\ ] \begin{bmatrix} i\tilde{\boldsymbol{g}}_B^T \\ \bar{\boldsymbol{L}}_B^T \end{bmatrix} \qquad \text{where } i^2 = -1;$$

then, almost the same arguments as before hold, i.e. by a sweep of *complex* Givens matrices



where $c^2 - s^2 = 1$, it is possible to obtain

$$\begin{bmatrix} \mathbf{0} & \tilde{\boldsymbol{L}}_{B'} \\ 0 & \tilde{\boldsymbol{l}}_h^T \\ 0 & \tilde{\boldsymbol{l}}_j^T \\ i\sigma_1 & \tilde{z}_1 \\ i\sigma_2 & \tilde{z}_2 \end{bmatrix}.$$

Note that, while $\tilde{\boldsymbol{g}}_B$ is nullified, all the "imaginary parts" are kept on the leftmost column alone, so that the resulting $\bar{\boldsymbol{L}}_B$ and $\tilde{\boldsymbol{l}}_h$, $\tilde{\boldsymbol{l}}_j$, $\tilde{z}_1$, $\tilde{z}_2$ are real matrix/vectors which bear many of the expected properties, such as $\tilde{z}_1\tilde{z}_2 = z_1z_2 - \sigma_1\sigma_2$ (and the same for the other scalar products): making the data structures "consistent", however, is somewhat more complicated that in the previous case, since zeroes along the diagonal of $\tilde{\boldsymbol{L}}_{B'}$ may be generated (but $\delta_h = \delta_j = 0$, i.e. no "new linear independence" is created). Dealing with such a situation is easy if it involves the last row of $\bar{\boldsymbol{L}}_B$, i.e. if after applying $b - 1$ (complex) Givens rotations the result is

$$\tilde{\boldsymbol{L}}_B = \begin{bmatrix} \mathbf{0} & \tilde{\boldsymbol{L}}_{B'} & \mathbf{0} \\ i\delta_h & \tilde{\boldsymbol{l}}_h^T & \tilde{l}_{hh} \end{bmatrix} \qquad \text{where} \qquad \tilde{l}_{hh}^{\,2} = \delta_h^2$$

so that no complex Givens matrix nullifying $i\delta_h$ can be found; however, it is also true that

$$\tilde{\boldsymbol{L}}_B = \begin{bmatrix} \mathbf{0} & \tilde{\boldsymbol{L}}_{B'} & \mathbf{0} \\ 0 & \tilde{\boldsymbol{l}}_h^T & 0 \end{bmatrix}$$

i.e. by simply "marking" h as linearly dependent a consistent situation is naturally recovered. The above relation holds true even if B already contains linearly dependent items, regardless of h being or not one of them, even though if the third dependent item is created then one of them must be removed from B (an O(1) task): if $\tilde{l}_{hh}^{\,2} = \delta_h^2$ for some h in position $k < b$, then with a sweep of (at most) $b - k - 1$ Givens rotations - those of Proposition I.3.2.8 - and a permutation of B the h-th item can be put in the last position, and the method applied.

As in §I.32, a set of cheap operations are available to modify the data of the problem, while keeping the data structures consistent and ready for a "warm" restart: this will be used in the next paragraph to efficiently cope with box constraints on the dual space.

I.3.5 Handling box constraints on the dual space

In some applications, (most of) the constraints $\boldsymbol{h} \geq \boldsymbol{yH}$ are just "boxes" $l_i \leq d_i \leq u_i$: in this paragraph, a version of TT that efficiently handles lower bounds $l_i \leq d_i$ on a subset γ of the variables is presented, and the extension to the more general class is straightforward. The BTT method still uses an *active set* strategy, in which at each step only a subset $\chi \subseteq \gamma$ of the primal multipliers associated to the box constraints are allowed to be nonzero: since the part of the base concerning subgradients (or general linear constraints) will be "implicitly" taken into account, to simplify the notation the subscript β for the columns of $\boldsymbol{G}$ will be dropped, and it will be considered partitioned as

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_\chi \\ \boldsymbol{G}_\eta \end{bmatrix}$$

where $\eta = \{ 1 .. n \} / \chi$ - that is, subscripts will hereafter indicate rows instead.
Given a "partial" base χ, the *restricted* problems to be solved at each step are

$$(\Pi_\chi) \qquad \min_{d,v} \left\{ v + 1/2 \left\| \boldsymbol{d} \right\|^2 : v\boldsymbol{e} \geq \boldsymbol{d}\boldsymbol{G} - \alpha \, , \boldsymbol{l}_\chi \leq \boldsymbol{d}_\chi \right\}$$

$$(\Delta_\chi) \qquad \min_{\theta,\mu} \left\{ \frac{1}{2} \left\| \begin{bmatrix} \boldsymbol{G}_\chi & -\boldsymbol{I} \\ \boldsymbol{G}_\eta & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \theta \\ \mu_\chi \end{bmatrix} \right\|^2 + \begin{bmatrix} \alpha & -\boldsymbol{l}_\chi \end{bmatrix} \begin{bmatrix} \theta \\ \mu_\chi \end{bmatrix} : \theta \in \Theta \, , \, \mu_\chi \geq \boldsymbol{0} \right\}$$

and therefore the "extended" Hessian matrix w.r.t. the "extended base" $\bar{B} = \{ 1 \ldots b \} \cup \chi$ is

$$\boldsymbol{Q}_{\bar{B}} = \begin{bmatrix} \boldsymbol{G}_\chi^T & \boldsymbol{G}_\eta^T \\ -\boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{G}_\chi & -\boldsymbol{I} \\ \boldsymbol{G}_\eta & \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{G}^T\boldsymbol{G} & -\boldsymbol{G}_\chi^T \\ -\boldsymbol{G}_\chi & \boldsymbol{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q} & -\boldsymbol{G}_\chi^T \\ -\boldsymbol{G}_\chi & \boldsymbol{I} \end{bmatrix};$$

a feasible solution of (Δ_χ) is optimal if and only if it solves the (KT) system

$$\begin{bmatrix} \boldsymbol{Q} & -\boldsymbol{G}_\chi^T \\ -\boldsymbol{G}_\chi & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \theta \\ \mu_\chi \end{bmatrix} = \begin{bmatrix} \rho\boldsymbol{e} - \alpha \\ \boldsymbol{l}_\chi \end{bmatrix} \qquad \text{for some } \rho \in \mathcal{R}$$

so that $\mu_\chi = \boldsymbol{G}_\chi\theta + \boldsymbol{l}_\chi$, and therefore $[\boldsymbol{G}^T\boldsymbol{G} - \boldsymbol{G}_\chi^T\boldsymbol{G}_\chi]\theta = \rho\boldsymbol{e} - [\alpha - \boldsymbol{G}_\chi^T\boldsymbol{l}_\chi]$. Since $\boldsymbol{G}^T\boldsymbol{G} - \boldsymbol{G}_\chi^T\boldsymbol{G}_\chi = \boldsymbol{G}_\eta^T \boldsymbol{G}_\eta = \boldsymbol{Q}_\eta$, the *relaxed* problem

$$(R_\chi) \qquad \min_{\theta,\mu} \left\{ \frac{1}{2} \left\| \begin{bmatrix} \boldsymbol{G}_\chi & -\boldsymbol{I} \\ \boldsymbol{G}_\eta & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \theta \\ \mu_\chi \end{bmatrix} \right\|^2 + \begin{bmatrix} \alpha & -\boldsymbol{l}_\chi \end{bmatrix} \begin{bmatrix} \theta \\ \mu_\chi \end{bmatrix} : \theta \in \Theta \right\}$$

(in which the constraints $\mu_\chi \geq \boldsymbol{0}$ are disregarded) is equivalent to the *unconstrained* problem

$$(\Delta^\eta) \qquad \min_\theta \left\{ 1/2\theta^T\boldsymbol{Q}_\eta\theta + \bar{\alpha}\theta : \theta \in \Theta \right\}$$

where $\bar{\alpha} = \alpha - \boldsymbol{G}_\chi^T\boldsymbol{l}_\chi$ and all the entries of the $\boldsymbol{g}_h$s corresponding to active constraints (whose indices are in χ) have been deleted. Therefore, the box-constrained problem (Π_γ) could be solved by using *any* algorithm for the unconstrained case as a "black box" to find a base B and a solution $\theta = [\theta_B \; \boldsymbol{0}]$ such that $[\theta \; \mu_\chi]$ solves the (KT) system for (Δ_χ): clearly, the efficient TT algorithm will be used as the subroutine TT(χ) that reports an optimal solution θ of (Δ_χ). Building on this, a "two-level" solver for the box constrained case can be constructed, where the lower level handles the θ variables while the upper level handles the μ variables: it should be clear, by just looking at the pseudo-code of Figure 15, that BTT can be viewed as TT applied to (Δ_γ), with just a different way of solving the (KT) system and a particular ordering in which constraints and subgradients are chosen for being inserted to and deleted from the "extended base" $\bar{B} = B \cup \chi$. In fact, convergence can be shown exactly in the same way - there cannot be infinitely many inner iterations, and every main iteration ensures a strict decrease - and all the data structures relative to a base B of the subproblem (Δ^η) have a straightforward interpretation in terms of the base $\bar{B}$ of the original problem (Δ_γ): for instance

$$[\theta^T \; \mu_\chi^T] \begin{bmatrix} \boldsymbol{Q} & -\boldsymbol{G}_\chi^T \\ -\boldsymbol{G}_\chi & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \theta \\ \mu_\chi \end{bmatrix} = \theta^T\boldsymbol{Q}_\eta\theta + \left\| \boldsymbol{l} \right\|^2 \quad \text{and} \quad [\alpha \; -\boldsymbol{l}_\chi] \begin{bmatrix} \theta \\ \mu_\chi \end{bmatrix} = \bar{\alpha}\theta - \left\| \boldsymbol{l} \right\|^2$$

and $\boldsymbol{d} = -[\, \boldsymbol{l}_\chi \; \boldsymbol{G}_\eta \theta \,]$, therefore (by letting v', $\boldsymbol{d}'$ and $f'()$ be relative to (Δ^η))

$$f_{\tilde{B}}([\, \theta_B \;\; \mu_\chi \,]) = f'_B(\theta_B) \qquad v = v' \qquad \boldsymbol{d} = [\, \boldsymbol{l}_\chi \; \boldsymbol{d}' \,] \qquad \boldsymbol{G}^T\boldsymbol{d} = \boldsymbol{G}_\eta^T\boldsymbol{d}' - \boldsymbol{G}_\chi^T\boldsymbol{l}_\chi.$$

This "hierarchic" view of the method greatly simplifies the design and the implementation of the algorithm but, due to the sophisticated reoptimization strategies discussed in §I.3.2 and §I.3.4, performances can be expected to be even better than those of the "monolithic" approaches known in the literature: obviously, the method immediately extends to $u_i \geq d_i$ constraints (where $\mu_i = -u_i - \boldsymbol{G}_i\theta$) and to general boxes $l_i \leq d_i \leq u_i$, since the two constraints $d_i \leq u_i$ and $l_i \leq d_i$ cannot be in the active set χ at the same time.

BTT Algorithm

```
χ = ∅; θ = TT( χ ); μ_χ = G_χ θ + l_χ;                 /* initialization */
while( ∃ i ∈ ( γ / χ ) such that l_i > − G_i θ )        /* main iteration */
    χ = χ ∪ { i };                                     /* item insertion − μ_i = G_i θ + l_i > 0 */
    do                                                 /* inner iteration */
        θ = TT( χ ); μ̄_χ = G_χ θ + l_χ;                /* KT solution - use TT as subroutine */
        if( ∃ i ∈ χ such that l_i < − G_i θ )          /* μ̄_i = G_i θ + l_i < 0 */
            then  w_χ = μ̄_χ − μ_χ;                     /* w_χ is of descent in the μ_χ space */
                  η = min_{i∈χ} { −w_i / μ_i : w_i < 0 }; /* maximum feasible step η > 0 */
                  μ_χ = μ_χ + η w_χ;                   /* take step η along w_χ */
            else  μ_χ = μ̄_χ; w_χ = 0;                  /* μ̄_χ is optimal for (Δ_χ) */
        forall( i such that μ_i = 0 )
            χ = χ / { i };                             /* item(s) deletion */
    while( w_χ ≠ 0 );                                  /* end inner iteration */
endwhile                                               /* end main iteration */
```

Figure 15: the BTT algorithm

A point still needs clarifications: the (expensive) matrix-vector product $\boldsymbol{G}_\chi^T\boldsymbol{l}_\chi$ need not be calculated from scratch at every iteration, since e.g. when $\chi' = \chi \cup \{ i \}$ one has

$$\bar{\alpha}' = \alpha + \boldsymbol{G}_\chi^T\boldsymbol{l}_{\chi'} = \alpha - [\, \boldsymbol{G}_\chi^T \;\; \tilde{\boldsymbol{g}} \,]\begin{bmatrix} \boldsymbol{l}_\chi \\ l_i \end{bmatrix} = \bar{\alpha} - l_i\tilde{\boldsymbol{g}}$$

where $\tilde{\boldsymbol{g}} = \boldsymbol{G}_i$ is the i-th row of $\boldsymbol{G}$; furthermore, the vector $z_2 = \boldsymbol{L}_{B'}^{-1}\bar{\alpha}$ can be updated without resorting to an expensive backsolve on $\boldsymbol{L}_{B'}$, as

$$\tilde{z_2} = \tilde{\boldsymbol{L}}_{B'}^{-1}\bar{\alpha}' = \tilde{\boldsymbol{L}}_{B'}^{-1}\bar{\alpha} - l_i[\; \tilde{\boldsymbol{L}}_{B'}^{-1}\tilde{\boldsymbol{g}} \;].$$

Actually, the method discussed in §I.3.4 finds $\tilde{\boldsymbol{L}}_{B'}^{-1}\bar{\alpha}$ instead, but it is easy to recover the "perturbation" vector $\tilde{\boldsymbol{L}}_{B'}^{-1}\tilde{\boldsymbol{g}}$, since the orthogonal $(b+1)\times(b+1)$ matrix $\boldsymbol{G}_S$ such that

$$[\; \tilde{\boldsymbol{g}} \;\; \boldsymbol{L}_{B'} \,]\boldsymbol{G}_S = [\; \boldsymbol{0} \;\; \tilde{\boldsymbol{L}}_{B'} \,] \qquad \Rightarrow \qquad [\; \tilde{\boldsymbol{g}} \;\; \boldsymbol{L}_{B'} \,] = [\; \boldsymbol{0} \;\; \tilde{\boldsymbol{L}}_{B'} \,]\boldsymbol{G}_S^T$$

is at hand: therefore, by letting $[\, \upsilon_0 \; \upsilon^T \,] = [\, 1 \;\; \boldsymbol{0}^T \,]\boldsymbol{G}_S$ (this vector can be found in linear time)

$$\tilde{\boldsymbol{L}}_{B'}\upsilon = [\; \boldsymbol{0} \;\; \tilde{\boldsymbol{L}}_{B'} \,]\begin{bmatrix} \upsilon_0 \\ \upsilon \end{bmatrix} = [\; \boldsymbol{0} \;\; \tilde{\boldsymbol{L}}_{B'} \,](\boldsymbol{G}_S^T)\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} = [\; \tilde{\boldsymbol{g}} \;\; \boldsymbol{L}_{B'} \,]\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} = \tilde{\boldsymbol{g}}$$

i.e. $\upsilon = \tilde{\boldsymbol{L}}_{B'}^{-1}\tilde{\boldsymbol{g}}$ and $\tilde{z}_2$ can be cheapily obtained from $\tilde{\boldsymbol{L}}_{B'}^{-1}\bar{\alpha}$. The case $\chi' = \chi / \{ i \}$ is analogous, with $\bar{\alpha}' = \alpha + l_i\tilde{\boldsymbol{g}}$ and $[\ \upsilon_0\ \upsilon^T\ ] = [\ -i\ \ \mathbf{0}^T\ ]\boldsymbol{G}_S$, and all the elements that are necessary to use TT as a subroutine for BTT have been introduced: a number of further details can be easily derived from the above theory, such as how to keep the scalar product updated ($z_1\tilde{z}_2 = z_1z_2 \pm \upsilon_0\sigma_1 l_i$ for σ_1 defined in §I.3.4), how to update $\boldsymbol{l}_\chi$ when moving "along" $\boldsymbol{d}$ of a step τ ($\boldsymbol{l}_\chi := (1 - \tau / t)\boldsymbol{l}_\chi$) or that the above results hold true for the generalised problem of §I.3.3 with constraint $\boldsymbol{c}\theta = 1$. The two-level structure is also important in the design of an efficient implementation of the BTT algorithm, that will be discussed in the next paragraph.

I.3.6 Implementation details

The implementation of the TT algorithm was the initial step of the larger project aimed at developing efficient Bundle codes for solving Lagrangean Duals of large-scale (Integer) Linear Problems: hence, an effort was made from the beginning to obtain a code that was not only efficient and easily usable, but also customizable to applications that might not have been foreseen at the developing time. This approach was very useful for the development of the different Bundle codes that have been produced so far, but it was also the basis for the ideation of the BTT algorithm - that was later imposed by the need of solving constrained NDO problems - and hence played a considerable role in all the research. Like all the other codes, the code is written in C++ and makes extensive use of its *encapsulation* capabilities, in that the whole TT algorithm is embedded in one single *object*, `class MinQuad`, with an abstract interface containing the (only) *methods* allowing the NDO algorithm to modify and view the data structures (β, $\boldsymbol{Q}_\beta$, α_β, t ...) and to perform the actual calculations. Moreover, the *inheritance* and *polymorphism* capabilities of the language are exploited since the BTT algorithm is contained in the *derived* `class BMinQuad`, inheriting from the *base class* most of its data and code, and only redefining the necessary methods through the mechanism of *virtual functions*.

Although a full description of the classes is not possible here, a brief review of the main methods of their interface may help a better understanding of the structure: the methods can be divided into

- a set of *public* methods for handling the data of (Δ_β), such as `AddSubGradient(`i, α_i`)` to add ($\boldsymbol{g}_i$, α_i) to β (and an analogous `AddConstraint()`), `ResetBundle(`i`)` to remove it, `ChangeAlfa(`α`)` or `MoveAlongD(`Δ_o, τ`)` for performing "movements" in the dual space and `Add[Remove]Variable(`$\tilde{\boldsymbol{g}}$`)` to control the variables of (Π_β); only a few of them need to be redefined by `BMinQuad`, that adds methods such as `Add[/Remove]ConstrVar(`i, l_i`)` or `MakeVar[Un]Constr(`i`)` to control the set of constrained variables. Although apparently simple, these methods implement the mechanisms discussed in §I.3.2 and §I.3.4 for allowing a fast reoptimization after a data change: furthermore, they isolate the NDO code from the details of the data structure implementation in the `MinQuad` and `BMinQuad` classes, that can therefore change (and they did) without requiring any modification in the NDO code. As an example of the usefulness of such an approach, the computational comparisons reported in the next paragraph have been obtained by simply "disguising" the two other (completely different) (QP) codes under the same interface, and keeping the two different NDO solvers untouched;
- the "main" method `MQFrontEnd(`t`)` for solving the current instance of (Δ_β) (with the given t): far

from being only a call to the actual core solver code (the *protected* `CalculateOptDir()`), this method implements a "dynamic" relative tolerances adjustment method that greatly improves resistance of the algorithm to ill-conditioned instances and to cumulation of rounding errors. In the TT algorithm, some "$y = 0$" tests that are crucial to determine the behaviour of the algorithm - such as the choice between cases (0), (1) and (2) and between "normal" and "infinite" steps - are implemented as $| y | < m\sigma\mu(\boldsymbol{L}_{B'})\varepsilon_R$, where $\mu(\cdot)$ is the condition number, σ is related to the "size" of the numbers used in the calculation and ε_R is the "dynamic" relative precision required: erroneous conditions (such as a $\delta << 0$ in I.3.2.6) forcing `CalculateOptDir()` to quit are interpreted as "increase ε_R" or "decrease ε_R" requests, so that ε_R is dynamically adjusted in order to face "bad" instances without a need for hand-tuning of the parameters. From-scratch recalculation of the main data structures ($\boldsymbol{L}_{B'}$, z_1 ...), that must always be implemented in any numerical code, is performed by the *protected* method `RebuildBase()` within the above ε_R tuning mechanism;

– methods for retrieving the solution informations like (B, θ_B) = `ReadMult()`, $\boldsymbol{dG}$ = `ReadGTd()` and - since the handling of the α vector can be entirely demanded to the class - α = `ReadAlfa()`; the `BMinQuad` class, where $\boldsymbol{d}_\chi$ has to be computed at each step, adds `ReadD()`, `ReadZ()` and related methods. As shown in Chapter I.2, sensitivity analysis information may also be useful to NDO solvers, and the method (l_1, l_2, l_3) = `SensitAnals1()` provides a full description of the optimal solutions values of (Δ_β) and (Π_β) as a function of t

$$\left\| \boldsymbol{G}\theta(t) \right\|^2 = \left\| \boldsymbol{d}(t) \right\|^2 = l_1 + (1 / t)^2 l_3$$

$$\sigma(t) = \alpha\theta(t) = l_2 - (1 / t) l_3$$

$$v(t) = - l_1 - (1 / t) l_2$$

at essentially no cost, since it is easy to verify that

	Case (0)	Case (1)
$l_1 (\geq 0)$	$1 / \left\| z_1 \right\|^2$	0
l_2	$z_1 z_2 / \left\| z_1 \right\|^2$	$\rho = (\alpha_h - \boldsymbol{l}_h z_2) / (1 - \boldsymbol{l}_h z_1)$
$l_3 (\geq 0)$	$[\left\| z_1 \right\|^2 \left\| z_2 \right\|^2 - (z_1 z_2)^2] / \left\| z_1 \right\|^2$	$\left\| \rho z_1 - z_2 \right\|^2 + \left\| \boldsymbol{l}_\chi \right\|^2$

and all the necessary data is either already calculated during the algorithm, or can be kept updated in O(1) per iteration. Other more costly information, such as the "stability interval" $t_m \leq t \leq t_M$ in which the current base B remains optimal (hence the above estimates on $\left\| \boldsymbol{d}(t) \right\|^2$, $\sigma(t)$ and $v(t)$ are "exact") and the two vectors θ_1, θ_2 such that $\theta(t) = \theta_1 + (1 / t)\theta_2$ can be queried with (t_m, t_M) = `SensitAnals2()` and (θ_1, θ_2) = `SensitAnals3()`;

– the *protected* interface with the NDO solver, that needs to be a derived class: in fact, the *pure virtual* method q_{ij} = `GiTGj(i, j)` is the only way in which `MinQuad` obtains information about the Hessian $\boldsymbol{Q}$, and it has to be implemented by the derived class. This makes `MinQuad` independent from how subgradients are stored in the NDO solver, allowing easy exploitation of sparsity or known structural properties of the vectors $\boldsymbol{g}_h$ (and $\boldsymbol{h}_h$) and a great degree of freedom in the design of Bundle codes: since subgradients (and constraints) are only known to `MinQuad` by means of their "symbolic names" (indices), the NDO solver can maintain the matrix $\boldsymbol{G}$ in a row-wise or

column-wise fashion (or both), with a sparse or dense (or mixed) implementation and also have a complete control on details such as which entry in the vector represents which variable, with the possibility of even "restructuring" the data on line. The `BMinQuad` class requires more information, hence adds to its protected interface other pure virtual methods such as $\boldsymbol{G}_h$ = `GiTilde(`h`)` or z_h = `CalculateZh(`h`)`: these methods also must be implemented by the derived class, and since variables are yet referenced through "symbolic names" - that need not have any connection with the actual data layout - the NDO code have a complete freedom in the choice of the data structures implementation.

A graphical resume of the structure of the `MinQuad` and `BMinQuad` classes is presented in Figure 16, where arrows represent functional interactions and/or data exchange with the "outside world", thick lines inheritance relationships and dashed lines the use of the *virtual* mechanism to redefine a method in a drived class.

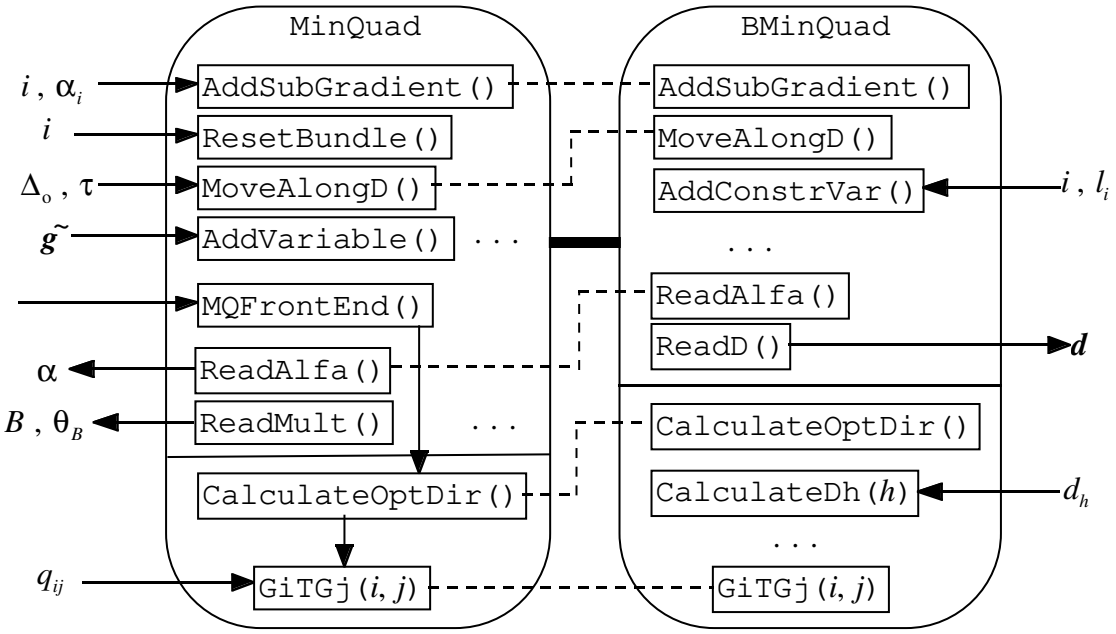


Figure 16: basic structure of the `MinQuad` and `BMinQuad` classes

A number of important advantages have been obtained with this structure: for instance, the `MinQuad` class alone can be used when no box constraints are required, avoiding any extra overhead, while only minor modifications in the NDO solver are required in order to use the more general `BMinQuad` instead; furthermore, code maintenance is much easier than if two separate codes had been written, and improvements in the TT code are automatically exploited by BTT - actually, as long as the interface remains the same TT, might also be replaced by an entirely different algorithm, and this possibility could be exploited in the future.

In the implementation of both classes, many different algorithmic options were available, among which the choice should have been dictated by the - unknown at developing time - characteristics of the "final" NDO code and, worse, of the particular set of instances: some of them are

– whether calculation of the new row $\boldsymbol{Q}_h$ have to be performed immediately after $\boldsymbol{g}_h$ is inserted, or each scalar product $\boldsymbol{g}_i^T\boldsymbol{g}_j$ have to be performed "on demand" the first time that the corresponding

element $\boldsymbol{Q}_{ij}$ is required;

- whether calculation of the whole direction $\boldsymbol{d}$ have to be entirely performed at the beginning of any inner iteration of the BTT algorithm, or each entry d_h have to be separately calculated "on demand";
- whether general box constraints $\boldsymbol{l} \le \boldsymbol{d} \le \boldsymbol{u}$ are required, or simple lower bounds $\boldsymbol{l} \le \boldsymbol{d}$ suffice;
- whether or not general linear constraints are required;
- whether the box bounds $(\boldsymbol{l}, \boldsymbol{u})$ will ever change during the lifetime of a `BMinQuad` object (but for the use of `MoveAlongD()`);
- which implementation of the $\boldsymbol{Q}$ matrix have to be chosen;
- whether the scalar products like $z_1 z_2$ have to be kept updated in O(1), or recalculated from scratch at every iteration to enhance numerical stability.

Since it was not practical (if ever possible) to accommodate all these different alternatives by means of the standard inheritance and polymorphism concepts, but a reasonable choice could not be done a priori, we resorted to the - much more C-ish - mechanism of "compile time switches": for each of these alternatives, a preprocessor variable is `#define`(d) in the include file of the class and appropriate pieces of code are compiled or not according to the value of the switch. Although being less elegant (once an alternative is chosen, it is fixed for all the instances of the class in that object code), this method have shown to be somewhat complementary to the use of different classes: in several cases, the correct choice of some of these implementative details was of great importance in obtaining good computational results.

The final noticeable detail of the [B]TT implementation is the use of a set of "abstract" basic data types that makes it "parametric" about the precision required: in particular, for all the different kinds of basic data algorithm (indices, entries of $\boldsymbol{L}$, entries of $\boldsymbol{g}_h$, entries of $\boldsymbol{Q}$, entries of $\boldsymbol{d}$ ...) a data type is defined (`Index`, `HpNum`, `SgNum`, `QuNum`, `LMNum` ...) and all the variables pertaining to that type are declared of this "abstract" data type rather than, say, `int` or `double`. This can be useful in several ways: for instance, the subgradients have often only small integer entries, so that their (long) scalar products may be computed with faster integer operations; furthermore, especially with small problems, it is possible to use single precision floating point numbers (confirming the robustness of our approach) rather than the usual double precision ones, and the integers corresponding to indices can be kept as small as the expected maximum size of the problems require. In a set of preliminary experiences, the correct choice of the right data types improved performances of 30% in time and of almost 50% in memory: some care is required, however, since the conversion between different formats is not costless, and the relative benefits of different choices can be very different on different target architectures. Furthermore, this mechanism made the porting of the codes easier, since the implementation of the basic C types can be very different on different machines: although portability is rarely a concern when implementing experimental codes, a little care in the design phase can greatly help to make the code widely usable on different situations and architectures.

I.3.7 Computational results and comparisons

Barring the "classical" (QP) algorithm in [Mi77], that solves the related but different (Δ_β^ε) problem, only two specialized algorithms for the unconstrained version of (Δ_β) [Ki86] [Mo87] have been proposed in the literature: the TT algorithm bears resemblance to both, but greatly enhances upon them in terms of cost per iteration. An algorithm for the more general (Δ_β) problem of §I.4.3 - for "separable" functions - is proposed in [Ki89]: using $m \times m$ matrices, however, it is admittedly not well-suited for the class of instances of interest here. Independently from our development, the recent [Ki94] proposes an algorithm that solves the same problem as [Ki89] avoiding the use of $m \times m$ matrices, so that its basic operations (inserting/deleting items in base, finding the descent direction …) have almost the same cost than in BTT: however, the latter improves on it on a number of important details, such as calculation of $f(\theta)$ in linear rather than quadratic time, lighter operations corresponding to the "downdate" the triangular factor after a constraint insertion (moving a newly created linear dependent item in last position, as described at the end of §I.3.4) and no row additions and subtractions needed at "basic variable changes" (since the constraint $\boldsymbol{e}\theta = 1$ is explicitly handled rather than "projected away" by replacing $\boldsymbol{Q}$ with $\hat{\boldsymbol{Q}} = \hat{\boldsymbol{G}}^T\hat{\boldsymbol{G}}$, where $\hat{\boldsymbol{G}} = [\, \boldsymbol{g}_1 - \boldsymbol{g}_h \ \ldots \ \boldsymbol{g}_m - \boldsymbol{g}_h \,]$ for some h). Furthermore, the two-level nature of BTT also helps, since whenever more than one step of the TT algorithm is performed at each inner iteration of BTT, recalculation of $\boldsymbol{d}$ (a "heavy" task) can be avoided; in general, performances can be improved by using different strategies at the two levels, since the same operation may have a different impact in the two cases: for instance, in TT the base must always be kept as small as possible, while this is not necessary in BTT. However, no implementation of the algorithm of [Ki94] was made available to us, and no computational experiences are reported (but it is considered efficient among the experts), so that no measure of relative efficiency could be obtained: therefore, [B]TT have been compared with two non specialized codes, to demonstrate at least the necessity of an ad-hoc approach. Several Bundle codes based on (early) implementations of the TT and the BTT algorithms have been developed in different contexts, such as Crew Scheduling problems [CGN95], Forestry problems [EGM95], Quadratic Semi-Assignment problems [MP95] and Multicommodity Flow problems [FG96]: hence, it was possible to compare the (QP) codes in a very "realistic" environment, by using them as (Δ_β)-solvers within some the above Bundle codes on "real" instances.

The first benchmark code is a "specialized" version of the QL code [Po83], that was modified to be used as (Δ_β)-solver within the Bundle code of [OSZ91]: the Fortran code was translated to C with the `f2c` converter by R. Freling and further adapted to C++ for the current experiments. Despite its age, it is well-known to be an efficient public domain (QP) solver: however, working on the "dual" (Π_β), and not being able to reoptimize from previous calls, it was expected to be slower than BTT. The second benchmark code is a beta release of the Interior Point (QP) solver of Cplex‡ 4.0, hereafter called CQP: it is based on the "modern" IP technology [CLM93], as opposed to the "old" active-set method of BTT, and exploits the Interior Point routines of one of the best commercial (LP) solvers on the market, using sophisticated sparse linear algebra routines and state-of-the-art coding. Hence, in this case it was not clear whether or not BTT would have really had a decisive advantage on it:

‡ Cplex is a registered trademark of Cplex Inc.

furthermore, since IP methods behave better on large problems with sparse Hessians such as (Π_β), that has a separable objective function, both the "primal" and "dual" formulation were solved.

Anticipating a bit the conclusions, [B]TT has shown to be blatantly faster than the other (QP) solvers: such a result is however limited to the specific application, and in no way has to be intended as a claim of superior efficiency outside this framework. Yet, a considerable care has been taken in order to make the tests fair: for instance, the cost of data structures initialisations and problem scaling was not counted, so that no advantage is given to BTT for its the problem-specific data structures; also, double precision arithmetic was used and the "compile time" options were kept fixed for all instances, even though savings could be obtained by exploiting its superior flexibility. Furthermore, the other codes were provided with all the "external" information that they could use: for instance, CQP was not asked to solve the "full" primal formulation (Δ_γ), but rather an easier restricted problem similar to (R_χ) w.r.t. the optimal set χ that had been previously determined by BTT.

The first set of experiences was performed with the Bundle code of [MP95] for finding Lagrangean-based lower bounds for the Quadratic Semi-Assignment Problem: the number of (unconstrained) Lagrangean multipliers varied, depending on the instance, from 10 up to 1600, and the maximal Bundle dimension reached 250 subgradients in the larger instances. For technical reasons, CQP and QL were tested against TT on two different (but analogous) sets of 50 and 130 instances respectively: the results are reported in Table 1 and Table 2, where *m* indicates the number of Lagrangean multipliers, *c* the number of calls to the (QP) solver and *b* the average Bundle dimension troughout the calls. Problems with similar "size" *mbc* are aggregated in groups (*n* indicates the number of instances), and the figures are averaged within the group: the chosen measure of "size" of an instance, actually corresponding to a large number of (Δ_β) instances, has shown to have the largest correlation with the running times. The performances of the (Δ_β)-solvers are measured in terms of the "total running time", i.e. the sum of the running times of all the (Δ_β)s - up to 1800 - needed to solve a single instance: times are in seconds on an HP9000-712/80 workstation with 64 Mb of RAM, the BTT and QL codes have been compiled with the HP C++ compiler (under HP-UX 9.05) invoking normal optimisations (−O), while CQP was only available as executable code.

n	*m*	*b*	*c*	*TT*	*QP*
6	12	15	128	0.02	0.16
12	15	17	175	0.03	0.36
17	70	30	289	0.16	2.58
25	75	43	502	0.62	9.89
20	94	50	701	1.38	27.59
19	153	88	1296	12.72	277.80
16	284	89	1531	9.50	295.29
11	686	108	1842	10.72	721.98
4	1506	111	1885	10.12	751.90

Table 1: comparison between TT and QL codes

n	*m*	*b*	*c*	*TT*	*QPP*	*QPD*
6	12	14	129	0.03	1.04	2.01
8	62	26	197	0.10	5.03	20.60
9	68	55	766	1.75	76.96	478.97
6	152	83	1269	8.91	346.91	2388.25
8	275	87	1327	11.60	590.91	2167.44
7	679	112	1821	10.37	2424.26	5022.14
6	1385	111	1874	10.00	2729.15	7607.03

Table 2: comparison between TT and CQP
(solving both the Primal and Dual formulation)

The results are better visualised in Chart 1 - Chart 4, and clearly show that to TT code is by far superior to both QL and CQP, especially when dealing with large problems (note that the scale in Chart 1 and Chart 3 is logarithmic): again, we remark that this is true only for this particular application, and with respect to the total running time over a whole execution of the NDO solver.

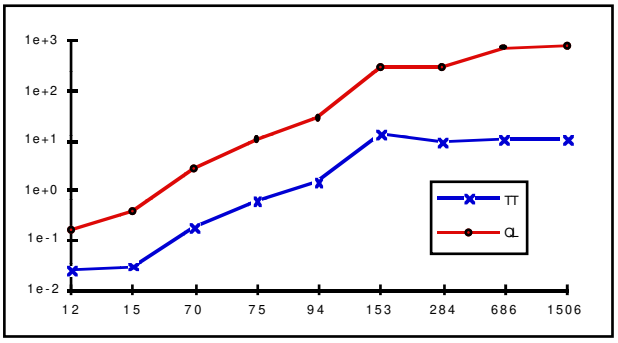


Chart 1: TT and QL running times

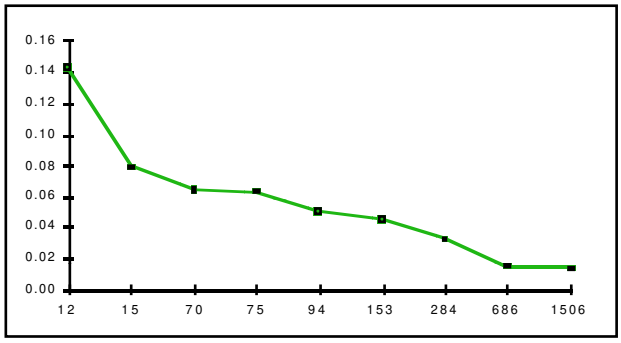


Chart 2: ratio between TT and QL running times

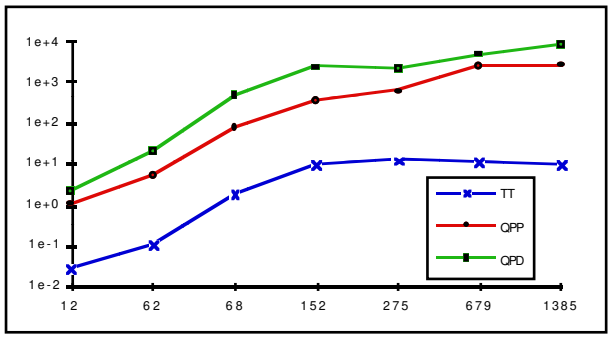


Chart 3: TT and CQP running times

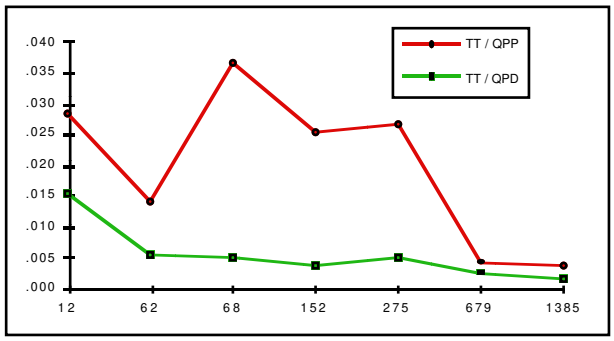


Chart 4: ratio between TT and CQP running times

The second comparison involved an early version of the MMCFB code, described in §II.2.1, ran on 43 Multicommodity Min Cost Flow (MMCF) problems (of the data set described in II.2.3.1): these instances are "easier" than the previous ones, since m (= number of edges in the graph) is between 200 and 700, and the Bundle code always terminated in less than 50 iterations - against up to 1000 - so that the average value of b was much smaller. Furthermore, from 94 to 99% of the nonnegativity constraints were always "active", and the average number of TT calls per BTT call were always smaller than 3 (and often ≈ 1 on small problems), so that the TT algorithm was never called more than 150 times - against up to 1800 - during the solution of any single problem: hence, this data set is representative of a somewhat different class of problems, in which reoptimization is possibly less important than the ability of rapidly solving small-sized instances. Due to limitations of the QL code at hand, only CQP was tested in this setting: the results are reported in Table 3, and show that the generic code is more than 25 times slower than BTT. On these problems, QPD outperforms QPP, even though, as shown in Chart 5 and Chart 6, for the largest instances the latter starts becoming competitive with the former. The results clearly show that the availability of an ad-hoc (QP) solver is crucial for a Bundle approach to be efficient in practice, at least in terms of total running time: other evidences supporting this assertion will be presented in Part II, but it should be already clear that not using a specialized code, i.e. paying 1-2 orders of magnitude more for the solution of (Δ_β), may well make a(n otherwise efficient) Bundle code unattractive w.r.t. alternative approaches.

n	*m*	*b*	*c*	*TT*	*QPP*	*QPD*
7	257	2	4	0.01	3.91	0.33
6	371	4	7	0.01	5.46	0.96
10	487	5	10	0.02	5.49	1.94
6	439	7	15	0.04	7.80	2.98
5	366	9	22	0.05	12.44	5.09
9	375	14	38	0.15	20.39	16.09

Table 3: comparison between BTT and CQP (solving both the Primal and Dual formulation)

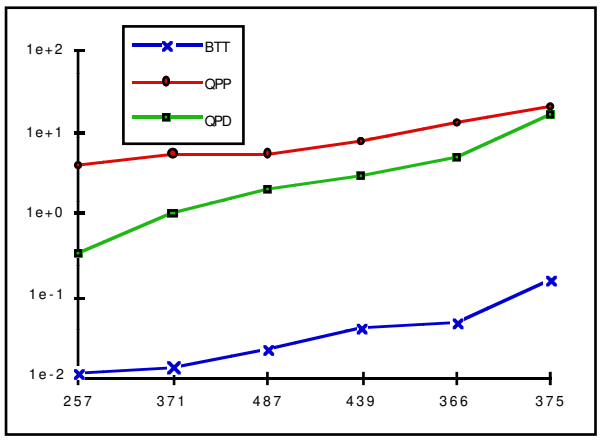


Chart 5: BTT and CQP running times

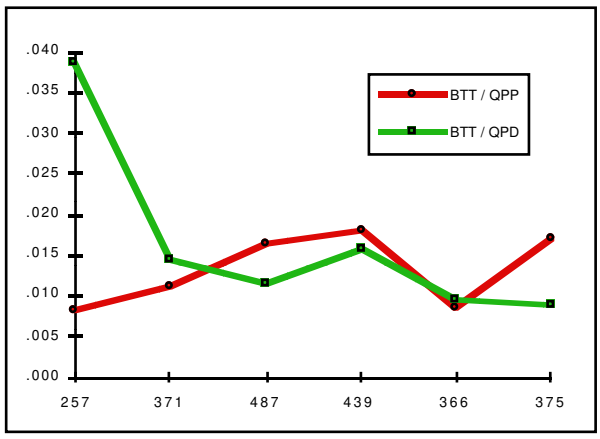


Chart 6: ratio between BTT and CQP running times

Another important feature of [B]TT, that makes it particularly well-suited as a (Δ_β)-solver, is that it allows the NDO solver to modify the dimension of the "dual" space at a low cost, allowing the implementation of *active-set* strategies that can further reduce the computational burden of the (Δ_β) solution. The importance of such strategies, that are clearly related to the multilevel pricing methods used in most simplex implementations, will be extensively proved in Part II: as a foretaste, we anticipate here some results obtained in the context of finding lower bounds to the Fixed Charge Multicommodity problem (Cf. §II.5.2). A sample of 12 medium-to-large size problems is presented in Table 4: each problem was solved twice, using (*Active Set*) or not (*Plain*) an active-set technique, but for the rest with exactly the same code and parameters. In the Table, the number of Lagrangean multipliers is reported in the *Variables* column: for both the variants, the number of (Δ_β) solved is reported in the *Calls* column (note that the Active Set strategy may change the number of steps performed by the Bundle algorithm), the total running time is reported in the *Total* column, and the time spent in the $\varphi(\boldsymbol{y})$ calculation is reported in the *Fi()* column.

	Active Set				Plain			
Variables	Calls	Total	Fi()	QP / Iter	Calls	Total	Fi()	QP / Iter
4400	178	24.24	9.17	0.08	186	67.04	7.56	0.32
4400	209	25.68	11.80	0.07	215	52.47	9.61	0.20
4400	100	13.05	5.38	0.08	98	37.76	4.13	0.34
4400	114	14.28	6.68	0.07	113	31.77	5.11	0.24
40400	501	2065.60	261.28	3.60	501	11688.60	257.08	22.82
40400	501	1041.37	244.90	1.59	501	4581.39	234.91	8.68
40400	501	1231.60	258.17	1.94	501	6062.63	239.96	11.62
40400	501	926.59	234.66	1.38	501	4137.74	228.24	7.80
6875	501	86.18	33.99	0.10	501	266.83	33.32	0.47
6875	347	68.24	26.60	0.12	347	161.64	25.75	0.39
6875	133	22.53	8.84	0.10	142	91.37	9.28	0.58
6875	129	22.24	9.25	0.10	134	73.93	9.32	0.48
63125	501	2410.37	456.92	3.90	501	10507.60	414.92	20.15
63125	501	2537.95	432.06	4.20	501	8291.95	370.12	15.81

Table 4: influence of the Active Set strategy on the overall efficiency

Wihout the Active Set strategy, the (Δ_β) solution takes from the 50% to the 98% of the total running time, especially on large problems: the Active Set strategy reduces the (Δ_β) cost per iteration of a 3 - 5

factor (Cf. the *QP/Iter* column), yielding a dramatic overall performance improvement. This is almost entirely due to the two-level structure of BTT, i.e. to its ability of fully exploiting both the active set of subgradients and the active set of variables from the previous call: although none of the sets is necessarily very "stable" among the iterations, they are stable enough to provide valuable information for helping in the (QP) solution.

Finally, a specific benefit of our C++ implementation is its simple and clear interface discussed in §I.3.6, that directly supports the basic operations of a Bundle code (insertion and deletion of subgradients and constraints, modification of α ...): it greatly simplified the task of developing and modifying the several specialized Bundle variants that have been realized so far, and isolated them from the (sometimes dramatic) changes that the implementation itself went through over the years.

I.4 (LP)-based Bundle methods

In §I.1.2, the primal view of Bundle methods has been presented in details: there, it has been shown that ($\Delta_{\beta t}$) is the "quadratic version" of the *Phase* 0 problem of the Dantzig-Wolfe approach

(M^p) $\quad \min_\theta \{ \| \boldsymbol{A}(\Sigma_j \boldsymbol{x}_j \theta_j) - \boldsymbol{b} \|_p : \theta \in \Theta \} \qquad p \in \{1, \infty\}$

with an extra linear term, weighted by $1 / t$, that takes into account the objective function value. In this Chapter, this idea will be further developed by introducing a new class of Bundle algorithms using (LP) subproblems instead of (QP) ones: they are presented in §I.4.1, where the differences and similarities with the "standard" case are discussed. Then, §I.4.2 adapts the convergence proof of Chapter I.2 to the new case: most of the results extends quite smoothly, also because the original presentation was tailored to this aim, but some need discussions and modifications. Finally, in §I.4.3 we present further extensions to the cases of *constrained* optimization and of *separable* functions, that arise in many practical applications such as those discussed in Part II.

I.4.1 Derivation

The discussion in Chapters I.1 and I.2 naturally leads to the idea of using the *linear direction finding subproblem*

($\Delta^p_{\beta t}$) $\quad \min_\theta \{ t\| \boldsymbol{G}_\beta \theta \|_p + \alpha_\beta \theta : \theta \in \Theta \} \; [+ \varphi(\bar{\boldsymbol{y}})]$

which has the same primal interpretations than ($\Delta_{\beta t}$), but more choices for the norm of the *primal stabilizing term*: for instance, ($\Delta^p_{\beta t}$) is the *restriction to* $\bar{X}_\beta$ of

($P^p_{\bar{y}, t}$) $\quad \min_{\boldsymbol{x}} \{ f^p_{\bar{y}, t}(\boldsymbol{x}) = (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\boldsymbol{x} + t\| \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \|_p : \boldsymbol{x} \in \bar{X} \} \; [+ \bar{\boldsymbol{y}}\boldsymbol{b}]$

which is a (generalized) *Augmented Lagrangean relaxation* of (P) (Cf. §I.2.1) with a (nonquadratic) nonlinear penalty term $t\| \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \|_p$. The L_1 and L_∞ norms are the two only practical alternatives to the Euclidean norm ($p = 2$): the explicit (LP) formulations of the corresponding problems are

($\Delta^1_{\beta t}$) $\quad \min_{\boldsymbol{x},\boldsymbol{w}} \{ t\boldsymbol{e}\boldsymbol{w} + \alpha\theta : -\boldsymbol{w} \le \boldsymbol{G}_\beta \theta \le \boldsymbol{w} \, , \theta \in \Theta \}$

($\Delta^\infty_{\beta t}$) $\quad \min_{\boldsymbol{x},w} \{ tw + \alpha\theta : -w\boldsymbol{e} \le \boldsymbol{G}_\beta \theta \le w\boldsymbol{e} \, , \theta \in \Theta \}$

where $\boldsymbol{e} = [1, 1, .. , 1]$ (note that $\boldsymbol{w}$ is a vector for $p = 1$, while w is a scalar for $p = \infty$). By elementary Linear Programming theory, their duals are

$(\Pi^1_{\beta t})$ $\max \{ v : v\boldsymbol{e} \le (\boldsymbol{d}^+ - \boldsymbol{d}^-)\boldsymbol{G}_\beta + \alpha_\beta , \boldsymbol{d}^+ + \boldsymbol{d}^- = t\boldsymbol{e} , [\boldsymbol{d}^+, \boldsymbol{d}^-] \ge \boldsymbol{0} \}$

$(\Pi^\infty_{\beta t})$ $\max \{ v : v\boldsymbol{e} \le (\boldsymbol{d}^+ - \boldsymbol{d}^-)\boldsymbol{G}_\beta + \alpha_\beta , \boldsymbol{d}^+\boldsymbol{e} + \boldsymbol{d}^-\boldsymbol{e} = t , [\boldsymbol{d}^+, \boldsymbol{d}^-] \ge \boldsymbol{0} \}$

where $\boldsymbol{d}^+$ and $\boldsymbol{d}^-$ are the dual variables of the $-\boldsymbol{w}$ $[-w\boldsymbol{e}] \le \boldsymbol{G}_\beta\theta$ and $\boldsymbol{G}_\beta\theta \le \boldsymbol{w}$ $[w\boldsymbol{e}]$ constraints respectively: under the obvious interpretation of $\boldsymbol{d}^+$ and $\boldsymbol{d}^-$ as "positive and negative" parts of the direction $\boldsymbol{d} = \boldsymbol{d}^+ - \boldsymbol{d}^-$, they can be rewritten as

$(\Pi^1_{\beta t})$ $\max_{d,v} \{ v : v \le \boldsymbol{d}\boldsymbol{g}_i + \alpha_i \;\; i \in \beta , \| \boldsymbol{d} \|_\infty \le t \} = \max_d \{ \varphi_\beta(\boldsymbol{d}) : \| \boldsymbol{d} \|_\infty \le t \}$

$(\Pi^\infty_{\beta t})$ $\max_{d,v} \{ v : v \le \boldsymbol{d}\boldsymbol{g}_i + \alpha_i \;\; i \in \beta , \| \boldsymbol{d} \|_1 \le t \} = \max_d \{ \varphi_\beta(\boldsymbol{d}) : \| \boldsymbol{d} \|_1 \le t \}$

i.e. the maximization of the CP-model on a "ball" (in a proper norm) of radius t around $\bar{\boldsymbol{y}}$. The *trust radious* nature of t is even clearer than in the $p = 2$ case: however, the new dual problems can still be written in the same *Proximal Point* fashion (Cf. §I.1.1) as $(\Pi^2_{\beta t})$

$(\Pi^p_{\beta t})$ $\max_d \{ \varphi_\beta(\boldsymbol{d}) + g^p_t(\boldsymbol{d}) \}$

where $g^p_t(\boldsymbol{d})$ is the concave *characteristic function* of the set $\{ \boldsymbol{d} : \| \boldsymbol{d} \|_q \le t \}$

$$g^p_t(\boldsymbol{d}) = \begin{cases} 0 & \text{if } \| \boldsymbol{d} \|_q \le t \\ -\infty & \text{otherwise} \end{cases}$$

for the proper choice of q, so that the "stabilizing" term of $(\Pi^2_{\beta t})$ transforming into "hard" constraints of $(\Pi^1_{\beta t})$ and $(\Pi^\infty_{\beta t})$ can be viewed as just using *exact* rather than *non-exact penalty functions* to keep the trial point inside a ball of radius t.

As simple as it looks, this idea have not received much attention in the past: no sophisticated convergence analysis seems to exist for the few related algorithms known in the literature, such as

- the old *Boxstep* method [Ma75], where t is kept fixed and the maximization over the box is performed exactly before moving the current point $\bar{\boldsymbol{y}}$: this can be viewed as a method where a "nonquadratic" Moreau-Yosida Regularization $\phi_t(\bar{\boldsymbol{y}})$ is exactly computed before attempting a step, and, not surprisingly, its performances are known to be very poor;
- [HL93 Algorithm 2.1.1], where the box is rather defined with the Euclidean norm: this does not appear to give any practical improvement on $(\Pi^2_{\beta t})$, and its convergence proof makes use of the *unicity* of the solution of the subproblem;
- the only other (independent) proposal for an (LP)-based NDO method [KCL95], where

  $(\underline{\Pi}^p_{\beta t})$ $\max_d \{ \varphi_\beta(\boldsymbol{d}) + t\| \boldsymbol{d} \|_p \}$

  is used as a "stabilized" version of (L_β): in the linear cases (p =1 and $p = \infty$), the dual problem

  $(\underline{\Delta}^p_{\beta t})$ $\min_\theta \{ \alpha\theta : \| \boldsymbol{G}_\beta\theta \|_q \le t , \theta \in \Theta \}$

  (for $q = \infty$ and $q = 1$ respectively), that is not described in the original treatment, is the direct generalization of (M^ε_β). Although the relative merits of the two approaches cannot be told without a computational comparison, $(\Delta^p_{\beta t})$ might be preferable to $(\underline{\Delta}^q_{\beta t})$ for the same reasons why $(\Delta_{\beta t})$ is preferable to (M^ε_β): in fact, the convergence proof of [KCL95] is rather weak and does not give any hint on how t should be handled, while in Chapter I.5 it will be shown that all the t-strategies

developed for $(\Delta^{2}_{\beta t})$ can be used for $(\Delta^{1}_{\beta t})$ and $(\Delta^{\infty}_{\beta t})$ as well.

The L_1 and L_∞ norms are "dual", i.e. if one is used in $(\Delta^{p}_{\beta t})$ for the "stabilizing" term, then the other is used in the Trust Region of $(\Pi^{p}_{\beta t})$, while the L_2 norm is "self-dual": a close analogy can be seen in the proof that the steepest ascent direction is the least-norm subgradient (Cf. §I.1.1), i.e.

$$(Q_{\bar{y}}) \qquad \operatorname{argmax}_{\boldsymbol{d}} \{ \varphi'(\bar{\boldsymbol{y}}, \boldsymbol{d}) : \| \boldsymbol{d} \| = 1 \} = \operatorname{argmin}_{\boldsymbol{g}} \{ \| \boldsymbol{g} \| : \boldsymbol{g} \in \partial\varphi(\bar{\boldsymbol{y}}) \}.$$

In fact, it is based on the relation

$$\varphi'(\bar{\boldsymbol{y}}, \boldsymbol{d}) = \min_{\boldsymbol{v}} \{ \boldsymbol{v}\boldsymbol{d} : \boldsymbol{v} \in \partial\varphi(\bar{\boldsymbol{y}}) \}$$

so that $$\max_{\boldsymbol{d}} \{ \varphi'(\bar{\boldsymbol{y}}, \boldsymbol{d}) : \| \boldsymbol{d} \| = 1 \} = \max_{\boldsymbol{d}} \{ \min_{\boldsymbol{v}} \{ \boldsymbol{v}\boldsymbol{d} : \boldsymbol{v} \in \partial\varphi(\bar{\boldsymbol{y}}) \} : \| \boldsymbol{d} \| = 1 \}$$

which, by Von Neumann's saddlepoint theorem [Ro70 p. 393], is equivalent to

$$\min_{\boldsymbol{v}} \{ \max_{\boldsymbol{d}} \{ \boldsymbol{v}\boldsymbol{d} : \| \boldsymbol{d} \| = 1 \} : \boldsymbol{v} \in \partial\varphi(\bar{\boldsymbol{y}}) \}$$

but for any fixed $\boldsymbol{v}$, the optimal $\boldsymbol{d}(\boldsymbol{v})$ is $(^{1}/_{\| \boldsymbol{v} \|})\boldsymbol{v}$ *because the intended norm is the Euclidean one*, yielding the result. However, it is also possible to use another "normalizing" norm (no pun intended) in the definition of steepest ascent direction, such as

$$(Q^{\infty}_{\bar{y}}) \qquad \max_{\boldsymbol{d}} \{ \min_{\boldsymbol{v}} \{ \boldsymbol{v}\boldsymbol{d} : \boldsymbol{v} \in \partial\varphi(\bar{\boldsymbol{y}}) \} : \| \boldsymbol{d} \|_\infty = 1 \}$$

and, reasoning as before, $(Q^{\infty}_{\bar{y}})$ is shown to be equivalent to

$$\min_{\boldsymbol{v}} \{ \max_{\boldsymbol{d}} \{ \boldsymbol{v}\boldsymbol{d} : -\boldsymbol{e} \le \boldsymbol{d} \le \boldsymbol{e} \} : \boldsymbol{v} \in \partial\varphi(\bar{\boldsymbol{y}}) \}$$

but now the optimal solution $\boldsymbol{d}(\boldsymbol{v})$ of the inner subproblem for fixed $\boldsymbol{v}$ is rather

$$d(\boldsymbol{v})_i = \begin{cases} 1 & \text{if } v_i > 0 \\ -1 & \text{if } v_i < 0 \\ 0 & \text{if } v_i = 0 \end{cases}$$

so that $\boldsymbol{v}\boldsymbol{d}(\boldsymbol{v}) = \| \boldsymbol{d} \|_1$ and therefore

$$\operatorname{argmax}_{\boldsymbol{d}} \{ \varphi'(\bar{\boldsymbol{y}}, \boldsymbol{d}) : \| \boldsymbol{d} \|_\infty = 1 \} = \operatorname{argmin}_{\boldsymbol{g}} \{ \| \boldsymbol{g} \|_1 : \boldsymbol{g} \in \partial\varphi(\bar{\boldsymbol{y}}) \}$$

with the same self-dual relationship between the L_1 and the L_∞ norms. Note that the above "L_1 steepest ascent direction" is in general *not unique*: even worse, it may not even be a direction of ascent, as shown in Figure 17 by all the directions comprised by the outer borders of the (filled) contour plot and the two dotted lines. Hence, (LP)-based direction finding problems are "weaker" than (QP)-based ones, and in fact the convergence theorems of §I.4.2 are weaker than the ones of Chapter I.2: nonetheless, it is possible to construct a theory that can accomplish for all the cases in an almost uniform way, disregarding as much as possible the specific choice of p. This is possible due to the many analogies between the direction finding problems, such as that, in all cases, $\boldsymbol{z}$ $(= \boldsymbol{G}_\beta\theta)$ is a $\sigma(= \alpha_\beta\theta)$-subgradient of $\varphi()$ in $\bar{\boldsymbol{y}}$, and since the Complementary Slackness conditions require

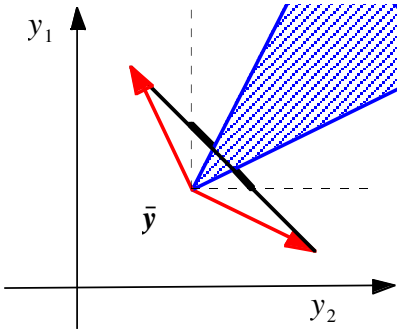


Figure 17: L_1 steepest ascent directions may *not* be of ascent

$$(\boldsymbol{d}\boldsymbol{G}_\beta + \alpha_\beta - v\boldsymbol{e})\theta = 0 \quad \Rightarrow \quad v = \boldsymbol{d}\boldsymbol{z} + \sigma \qquad (4)$$

$v = \varphi_\beta(\boldsymbol{d})$ is always the increase predicted by the CP-model, and also by the "aggregated CP-model" made of the only $\boldsymbol{z}$. Furthermore, although it is not in general true for $p \in \{1, \infty\}$ that $\boldsymbol{d} = t\boldsymbol{z}$ (unless $\bar{\boldsymbol{y}} + \boldsymbol{d}$ lies in the interior of the trust region), it holds for $p = 2$, so that the relations

$$\boldsymbol{dz} = t\| \boldsymbol{G}_\beta\theta \|_p \qquad \Rightarrow \qquad v = t\| \boldsymbol{G}_\beta\theta \|_p + \alpha_\beta\theta \tag{5}$$

can be thought to hold if the customary assumption to write $\|\cdot\|_2$ for $\|\cdot\|_2^2$ is respected; the *squared* norm in the $p = 2$ case also justifies some slight differences with the $p \in \{1, \infty\}$ cases, such as the extra $1/2$ factor in the objective function that is necessary in order to obtain the $v \le \boldsymbol{dg}_i + \alpha_i$ constraints in the dual. However, similarities are much deeper than differences: for instance, although the equalities $v = z(\Delta^p_{\beta t}) = z(\Pi^p_{\beta t})$ for $p \in \{1, \infty\}$ are no more satisfied for $p = 2$, the inequalities $1/2 v \le z(\Delta^2_{\beta t}) \le v \le 2z(\Delta^2_{\beta t})$ clearly hold, so that

$$z(\Delta^p_{\beta t}) \le \tau\| \boldsymbol{G}_\beta\theta \|_p + \alpha_\beta\theta \qquad \forall\, \tau > 0\,,\ \forall\, p \in \{1, 2, \infty\} \tag{6}$$

since θ is a feasible solution for $(\Delta^p_{\beta t})$, hence $z(\Delta^2_{\beta t}) \le 1/2\tau\| \boldsymbol{G}_\beta\theta \|_2^2 + \alpha_\beta\theta \le \tau\| \boldsymbol{G}_\beta\theta \|_2^2 + \alpha_\beta\theta$.

I.4.2 Convergence Properties

In this paragraph, the analysis of Chapter I.2 for the $p = 2$ case is extended to the linear cases: in a large part, this will only call for showing that the same relations hold despite the differences. Quite often, however, this would not even be needed: this is the case for instance of Observation I.2.1.1, Theorem I.2.3.1 and Observation I.2.3.4, that can be easily seen to hold - provided that all their prerequisites are shown to be true - independently from the choice of the norm. In some other cases, the equivalence simply derives from the fact that some relations are uniformly true: for instance (4), (5) and (6) can be used to show that $\| \boldsymbol{G}\theta^* \|_p = 0 \Rightarrow v^* = \varphi(\bar{\boldsymbol{y}} + \boldsymbol{d}^*) = z(\Pi^p_{t^*}) = z(\Delta^p_{t^*}) = \alpha\theta^*$ and $z(\Delta^p_{t^*}) \le \sigma + t^*\| \boldsymbol{z} \|_p$, which is all that is needed to extend Theorem I.2.1.2; besides, that theorem has a much clearer interpretation in the $p \in \{1, \infty\}$ case, since it tells that t^* had been properly chosen if, after convergence, an optimal solution $\boldsymbol{y}^*$ is "within reach" in the sense that $\| \bar{\boldsymbol{y}} - \boldsymbol{y}^* \|_q \le t^*$, while for $p = 2$ the condition is harder to state. Analogously, Theorem I.2.3.5 only uses (4) and (5), so that it is readily extended to the new setting, and Observation I.2.2.1 is obvious for $p \in \{1, \infty\}$, since $\| \boldsymbol{d} \|_q \le t$ (for $t = t_h$ and a proper norm q) are a part of the $(\Delta^p_{\beta t})$ constraints.

Conversely, Theorem I.2.2.2 does *not* hold true, but the idea that a sequence of consecutive NS is essentially an inner linearization approach applied to $(P^p_{\bar{y};\, t})$ is still valid: in fact, being θ and $\boldsymbol{d}$ the optimal solution of $(\Delta^p_{\beta t})$ and $(\Pi^p_{\beta t})$ respectively and $\bar{\boldsymbol{x}} = \Sigma_{j \in \beta}\, \boldsymbol{x}_j\theta_j$, the following theorem shows in this case what for $p = 2$ was just an easy consequence of the properties of the gradient of a differentiable function.

Theorem I.4.2.1: *$\boldsymbol{s} = \boldsymbol{c} - (\bar{\boldsymbol{y}} + \boldsymbol{d})\boldsymbol{A}$ is a subgradient of $f^p_t()$ in $\bar{\boldsymbol{x}}$; furthermore, it "cuts away" all the set $\bar{X}_\beta$, i.e. $\boldsymbol{sx} \ge \boldsymbol{s}\bar{\boldsymbol{x}}$ for each x belonging to $\bar{X}_\beta$.*

Proof: For any p, $\bar{\boldsymbol{x}}$ is the optimal solution of

$$(\Delta^p_{\beta t}) \qquad \min_{\boldsymbol{x}} \left\{ f^p_t(\boldsymbol{x}) : \boldsymbol{x} \in \bar{X}_\beta \right\}$$

and it is easy to check that $\nabla(f^2_t)(\bar{\boldsymbol{x}}) = \boldsymbol{s} = \boldsymbol{c} - (\bar{\boldsymbol{y}} + \boldsymbol{d})\boldsymbol{A}$; hence, for $p = 2$ the thesis follows from

the properties of the gradient of a differentiable function. Of the other cases, only $p = 1$ will be detailed, since the other is identical: the proof is based on the properties of the *perturbation* (otherwise called *value*) *function* of an (LP). Given the problem

$$\min_x \{ \boldsymbol{lx} : \boldsymbol{Mx} \le \boldsymbol{m} , \boldsymbol{Hx} \le \boldsymbol{h} \} \qquad (7)$$

its perturbation function w.r.t. the constraints $\boldsymbol{Mx} \le \boldsymbol{m}$ is

$$\bar{f}(z) = \min_x \{ \boldsymbol{lx} : \boldsymbol{Mx} \le \boldsymbol{m} + z , \boldsymbol{Hx} \le \boldsymbol{h} \}$$

and it is well-known [Ge69] that the dual optimal variables of (7) corresponding to the "perturbed" constraints $\boldsymbol{Mx} \le \boldsymbol{m}$ are a *subgradient of* $\bar{f}()$ *in* $\mathbf{0}$: hence, by choosing $z^+ = -\boldsymbol{A}z$ and $z^- = -\boldsymbol{A}z$ as the "perturbation vectors" of the $-\boldsymbol{w} \le \boldsymbol{G}_\beta\theta$ and $\boldsymbol{G}_\beta\theta \le \boldsymbol{w}$ constraints (with dual variables $\boldsymbol{d}^+$ and $\boldsymbol{d}^-$) respectively, one has

$$\bar{f}_\beta(z) = \min_{x,w} \{ \boldsymbol{tew} + (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\boldsymbol{x} : -\boldsymbol{w} \le \boldsymbol{b} - \boldsymbol{A}(\boldsymbol{x} + z) \le \boldsymbol{w} , \boldsymbol{x} \in \bar{X}_\beta \}$$

so that $[\boldsymbol{d}^+ , \boldsymbol{d}^-][-\boldsymbol{A} , \boldsymbol{A}] = -\boldsymbol{dA}$ is a subgradient of $\bar{f}()$ in $\mathbf{0}$. Therefore, $\boldsymbol{s} = \boldsymbol{c} - (\bar{\boldsymbol{y}} + \boldsymbol{d})\boldsymbol{A}$ is a subgradient of

$$\tilde{f}_\beta(z) = \bar{f}_\beta(z) + (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\boldsymbol{x} = \min_x \{ f^1(\boldsymbol{x} + z) , \boldsymbol{x} \in \bar{X}_\beta \}$$

in $z = \mathbf{0}$: this new function $\tilde{f}_\beta()$ has a clear interpretation in terms of the original $f^1()$, since it tells what the $z(\Delta^1_{\beta t})$ would be if the whole $\bar{X}_\beta$ would be translated along z; hence, it is reasonable that the new point is searched along $-\boldsymbol{s}$ (minimizing $\boldsymbol{sx}$), since $-\boldsymbol{s}$ is a (possible) direction of descent for $\tilde{f}_\beta()$, i.e. "enlarging" $\bar{X}_\beta$ in that direction (may) lead to a better value of $z(\Delta^1_{\beta t})$.

The proof that $\boldsymbol{s}$ is a subgradient of $f^1()$ in $\bar{\boldsymbol{x}}$ and that *all* the points of $\bar{X}_\beta$ have to lie in the half-space pointed by $\boldsymbol{s}$ with respect to $\bar{\boldsymbol{x}}$ can be illustrated by the following numerical example: consider the (Δ^1_i) problem where $\boldsymbol{b} = \mathbf{0}$, $\boldsymbol{c} = \mathbf{0}$, $\bar{\boldsymbol{y}} = \mathbf{0}$, $\boldsymbol{A} = -\boldsymbol{I}$, $t = 1$ and $X = \text{conv}\{ (1, 0), (3, 3), (3, -3) \}$; its dual (Π^1_i) requires maximization of the very simple function $\varphi()$, whose level sets are depicted in Figure 18, under the "box" constraints $\| \boldsymbol{d} \|_\infty \le 1$.

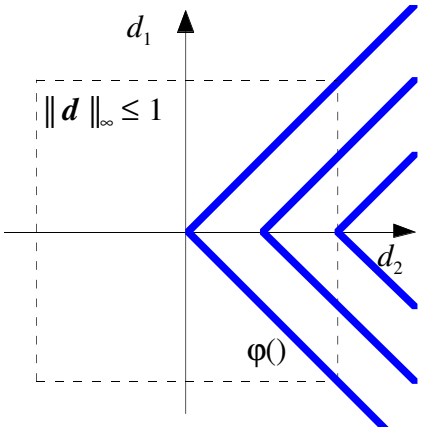


Figure 18: dual form of the example problem

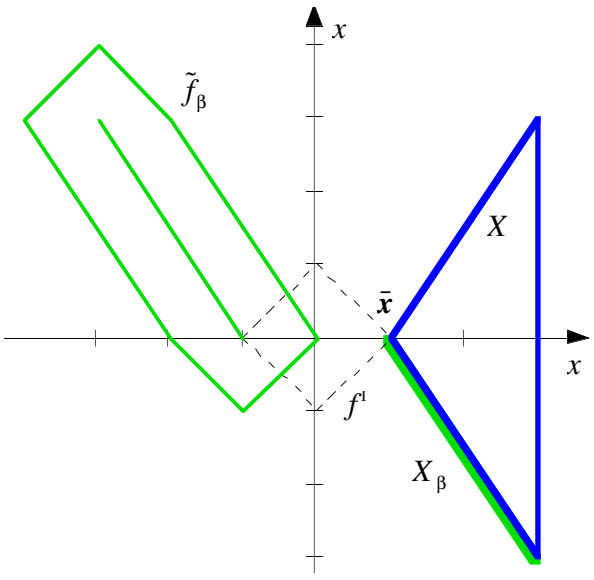


Figure 19: primal form of the example problem

Now, let $\beta = \{ (1, 0), (3, -3) \}$, and consider the corresponding $(\Delta^1_{\beta r})$; in Figure 19, X (= $\bar{X}$), X_β, the level set of $f^1()$ passing from the optimal solution $\bar{\boldsymbol{x}} = (1, 0)$ and some level sets of $\tilde{f}_\beta(\ \boldsymbol{x}\ )$ are shown together to group all the relevant information on the same picture. It is easy to see how the level set of $\tilde{f}_\beta()$ can be obtained by "moving" the original level set of $f^1()$ along (a specular image of) X_β, so that all the subgradients of $\tilde{f}_\beta()$ in $\mathbf{0}$ are subgradients of $f^1()$ in $\bar{\boldsymbol{x}}$; furthermore, $\partial \tilde{f}_\beta(\ \mathbf{0}\ )$ is strictly contained in $\partial f^1(\ \bar{\boldsymbol{x}}\ )$, since the vector (1, 1), that does not "cut away" X_β from $\bar{\boldsymbol{x}}$, belongs to the latter but not to the former.

In general, for any $\boldsymbol{g} \in \partial \tilde{f}_\beta(\ \mathbf{0}\ )$, the definition of subgradient tells

$$\tilde{f}_\beta(\ \boldsymbol{z}\ ) \geq \tilde{f}_\beta(\ \mathbf{0}\ ) + \boldsymbol{g}(\ \boldsymbol{z} - \mathbf{0}\ ) \quad \Rightarrow \quad \min_{\boldsymbol{x}} \{\ f^1(\ \boldsymbol{x} + \boldsymbol{z}\ ),\ \boldsymbol{x} \in \bar{X}_\beta\ \} \geq f^1(\ \bar{\boldsymbol{x}}\ ) + \boldsymbol{g}\boldsymbol{z}$$

so for any $\boldsymbol{x}'$, by letting $\boldsymbol{z} = \boldsymbol{x}' - \bar{\boldsymbol{x}}$ and noticing that $\tilde{f}_\beta(\ \boldsymbol{x}' - \bar{\boldsymbol{x}}\ ) \leq f^1(\ \boldsymbol{x}'\ )$, one has

$$f^1(\ \boldsymbol{x}'\ ) \geq f^1(\ \bar{\boldsymbol{x}}\ ) + \boldsymbol{g}(\ \boldsymbol{x}' - \bar{\boldsymbol{x}}\ ) \quad \Rightarrow \quad \boldsymbol{g} \in \partial f^1(\ \bar{\boldsymbol{x}}\ ).$$

Furthermore, take any $\boldsymbol{g} \in \partial f^1(\ \bar{\boldsymbol{x}}\ )$ such that there exist $\boldsymbol{x}' \in \bar{X}_\beta$ with $\boldsymbol{g}(\ \boldsymbol{x}' - \bar{\boldsymbol{x}}\ ) < 0$: by letting $\boldsymbol{z} = \bar{\boldsymbol{x}} - \boldsymbol{x}'$, one has that $\tilde{f}_\beta(\ \boldsymbol{z}\ ) \leq \tilde{f}_\beta(\ \mathbf{0}\ ) = f^1(\ \bar{\boldsymbol{x}}\ )$ (take $\boldsymbol{x} = \boldsymbol{x}'$ in the definition of $\tilde{f}_\beta()$), so that

$$\tilde{f}_\beta(\ \boldsymbol{z}\ ) \leq \tilde{f}_\beta(\ \mathbf{0}\ ) < \tilde{f}_\beta(\ \mathbf{0}\ ) + \boldsymbol{g}(\ \boldsymbol{z} - \mathbf{0}\ ) \qquad (\text{ since } \boldsymbol{g}\boldsymbol{z} = \boldsymbol{g}(\ \bar{\boldsymbol{x}} - \boldsymbol{x}'\ ) > 0\ )$$

and $\boldsymbol{g}$ violates the definition of subgradient of $\tilde{f}_\beta()$ in $\mathbf{0}$, yielding the contradiction. ♦

The example inside Theorem I.4.2.1 can also be used to prove that heedless aggregation can lead to cycling: note that the situation depicted in Figure 19 could be that at the second step of a Bundle algorithm, since with $\boldsymbol{y} = \bar{\boldsymbol{y}} = \mathbf{0}$ any point of X is optimal for the pricing subproblem, hence $\boldsymbol{x}_1 = \bar{\boldsymbol{x}} = (1, 0)$ (which is by the way the optimum) could have been selected. Then, any vector in $\partial \tilde{f}_\beta(\ \mathbf{0}\ ) = \partial f^1(\ \bar{\boldsymbol{x}}\ ) = \text{conv}\{\ (1, 1), (1, -1)\ \}$ is a candidate for the search direction $\boldsymbol{s}$, and $\boldsymbol{d}_1 = (1, 1)$ could be chosen: the optimal solution of the pricing subproblem is now $\boldsymbol{x}_2 = (3, -3)$, yielding the situation above. The next optimal solution of $(\Pi^1_{\beta r})$ with $\beta = \{ (1, 0), (3, -3) \}$ is still $\bar{\boldsymbol{x}}$, corresponding to multipliers $\theta = (1, 0)$: however, the set of dual optimal solutions has changed (typically, $\boldsymbol{d}_1$ is no more optimal), so suppose $\boldsymbol{d}_2 = (1, -1)$ is selected, generating $\boldsymbol{x}_3 = \boldsymbol{g}_3 = (3, 3)$. But, aggregation uses the previous θ, hence does not use $\boldsymbol{x}_3$: eliminating $\boldsymbol{x}_2 = (3, -3)$ from the Bundle produces a symmetric situation to that of the previous step, making $\boldsymbol{d}_1$ again a possible optimal solution and giving the chance to the algorithm to cycle.

Note that, in the above example, some sort of "degeneracy" has played a role in that an "active" subgradient whit $\theta_i = 0$ was eliminated: $\boldsymbol{g}_i$ is active at the h-th step if it is binding at $\boldsymbol{y}_h$, i.e. such that $\alpha_i + \boldsymbol{g}_i\boldsymbol{d}_h = v_h = \varphi(\ \boldsymbol{y}_h\ ) - \varphi(\ \bar{\boldsymbol{y}}\ )$, corresponding from the primal side to $\boldsymbol{s}_{hr}\boldsymbol{x}_i = \boldsymbol{s}_{hr}\bar{\boldsymbol{x}}_h$, i.e. that $\boldsymbol{x}_i$ belongs to the face of $\bar{X}_\beta$ induced by $\boldsymbol{s}_h$ and passing from $\bar{\boldsymbol{x}}_h$. By the Complementary Slackness Conditions, every subgradient with $\theta_i > 0$ is active, while the converse is not true: hence, one may wonder if convergence could be ensured by requiring that no active subgradient is ever eliminated from β (a strengthened definition of

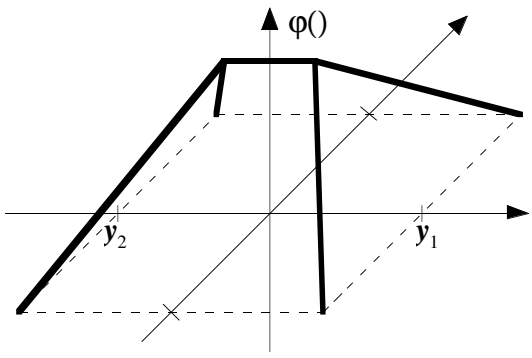


Figure 20: a further example of cycling

monotone β-strategy). Unfortunately, it is easy to construct a counterexample for this β-strategy also: on the function depicted in Figure 20, a Cutting Plane algorithm having the "front" and "rear" facets may cycle between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, each time acquiring the (respectively) "right" and "left" facets and discarding them at the following step because they are no more active.

Although theoretically relevant, all the above counterexamples sound very artificial: in practice, it is very unlikely that cycling may occur, since most of the real-world functions have an enormous number of facets. However, they clearly indicate that some differences, related with *uniqueness* of the subproblems solution, exist between the case $p = 2$ and $p \in \{1, \infty\}$. Nonetheless, Theorem I.4.2.1 is all that is needed to prove that, if an infinite sequence of NS is performed, the sequence of the optimal objective function values converges to $z(\mathrm{P}_t^p)$ (for some t that may not be known in advance), provided that β is allowed to grow indefinitely: hence, the next results will be stated under the (temporary) assumption that no subgradient inserted into β is ever removed until termination.

Theorem I.4.2.2: *if an infinite sequence of NS is performed,* $\{ f_i \} \rightarrow z(P_t^p)$ *for some* t*; furthermore, if* $m_1 < 1$ *then* $\{ v_i \} \rightarrow 0$ *and* $z(P_t^p) = z(P) = \varphi(\bar{\boldsymbol{y}})$.

Proof: Since *(iii)* applies, there must exist an h such that $t_i = t_h$ for all $i \geq h$, and the method will converge to an optimal solution of (P_t^p) for $t = t_h$: hence, it is possible to disregard whatever had happened before iteration h. Analogously, at each iteration i ($\geq h$) some new vertices of $\bar{X}$ are generated, but following *(v)* the vertex $\boldsymbol{x}_i$ produced during the i-th iteration will be the one corresponding to the evaluation of $\varphi(\bar{\boldsymbol{y}} + \boldsymbol{d}_i)$, i.e. a solution of the i-th *pricing subproblem*

$$(\mathrm{P}_i) \qquad \min_x \{ \boldsymbol{s}_i \boldsymbol{x} : \boldsymbol{x} \in X \}.$$

The subgradient inequality for $\boldsymbol{s}_i$ w.r.t. $f_t^p()$ at $\bar{\boldsymbol{x}}_i$ is

$$f_t^p(\boldsymbol{x}) \geq \underline{f}(\boldsymbol{x}) = f_t^p(\bar{\boldsymbol{x}}) + \boldsymbol{s}_i(\boldsymbol{x} - \bar{\boldsymbol{x}}_i) \quad \forall\, \boldsymbol{x}$$

so that, again, (P_i) can be seen as *relaxation* of (P_t^p) where the *linear lower approximation* $\underline{f}()$ is minimized instead of $f_t^p()$: hence, $\underline{f}(\boldsymbol{x}_i)$ is a *lower bound* on $z(\mathrm{P}_t^p)$, and therefore

$$f_t^p(\bar{\boldsymbol{x}}_i) \geq z(\mathrm{P}_t^p) \geq f_t^p(\bar{\boldsymbol{x}}_i) + \boldsymbol{s}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i) \quad \forall\, h \leq k < i.$$

The strong assumption on the β-strategy, i.e. the fact that β is a monotonically increasing set, together with Theorem I.4.2.1, allows to write

$$\boldsymbol{s}_t \boldsymbol{x}_k \geq \boldsymbol{s}_t \bar{\boldsymbol{x}}_i \qquad \forall\, h \leq k < i$$

that (changing the sign and adding $\boldsymbol{s}_t \boldsymbol{x}_i$ on both sides) turns to

$$0 \geq \boldsymbol{s}_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i) \geq \boldsymbol{s}_i(\boldsymbol{x}_i - \boldsymbol{x}_k) \qquad \forall\, h \leq k < i$$

and altogether gives

$$0 \geq z(\mathrm{P}_t^p) - f_t^p(\bar{\boldsymbol{x}}_i) \geq \max\{ \boldsymbol{s}_i(\boldsymbol{x}_i - \boldsymbol{x}_k) : h \leq k < i \}$$

but since $\{ \boldsymbol{x}_i \}$ is contained in the compact set X, a converging subsequence can be found: being $f_t^p()$ a Lipschitz function on X, $\| \boldsymbol{s}_i \|$ is bounded by some constant and for $i \rightarrow \infty$ the maximand goes to zero, showing that $\{ f_i = f_t^p(\bar{\boldsymbol{x}}_i) \} \rightarrow z(\mathrm{P}_t^p)$. Note that, theoretically speaking, if the cluster point $\hat{\boldsymbol{x}}$ of the $\{ \boldsymbol{x}_i \}$ sequence was known in advance, all the past $\boldsymbol{x}_i$ but for the nearest to $\hat{\boldsymbol{x}}$ could be discarded,

but since there's no way to know, everything must be kept: this rises the interesting question of whether some "measure of closeness" to $\hat{\boldsymbol{x}}$ could be obtained and used for keeping the dimension of β low. From the dual side, *(iv)* requires

$$\boldsymbol{g}_i\boldsymbol{d}_i + \alpha_i = \varphi(\boldsymbol{y}_i) - \varphi(\bar{\boldsymbol{y}}) < m_1 v_i \leq v_i = \boldsymbol{z}_i\boldsymbol{d}_i + \sigma_i$$

that (using $\alpha_i = (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\boldsymbol{x}_i - \varphi(\bar{\boldsymbol{y}}) + \bar{\boldsymbol{y}}\boldsymbol{b} \Rightarrow \sigma_i = (\boldsymbol{c} - \bar{\boldsymbol{y}}\boldsymbol{A})\bar{\boldsymbol{x}}_i - \varphi(\bar{\boldsymbol{y}}) + \bar{\boldsymbol{y}}\boldsymbol{b}$ and $\boldsymbol{g}_i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i \Rightarrow \boldsymbol{z}_i = \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}}_i$) implies

$$\begin{aligned} 0 \quad < \quad & v_i - \varphi(\boldsymbol{y}_i) + \varphi(\bar{\boldsymbol{y}}) \;=\; \boldsymbol{z}_i\boldsymbol{d}_i + \sigma_i - (\boldsymbol{g}_i\boldsymbol{d}_i + \alpha_i) \quad = \\ = \quad & (\boldsymbol{c} - (\bar{\boldsymbol{y}} + \boldsymbol{d}_i)\boldsymbol{A})\bar{\boldsymbol{x}}_i - (\boldsymbol{c} - (\bar{\boldsymbol{y}} + \boldsymbol{d}_i)\boldsymbol{A})\boldsymbol{x}_i \quad = \; \boldsymbol{s}_i(\bar{\boldsymbol{x}}_i - \boldsymbol{x}_i) \end{aligned}$$

but we have already shown that $\{ \boldsymbol{s}_i(\bar{\boldsymbol{x}}_i - \boldsymbol{x}_i) \} \to 0$, hence $\varphi(\boldsymbol{y}_i)$ becomes arbitrarily near to $v_i + \varphi(\bar{\boldsymbol{y}})$: if $m_1 < 1$, this in turn imply that $\{ v_i \} \to 0$, because otherwise a SS should have been performed at some time. As in the Theorem I.2.2.2, this allows to conclude $z(\mathrm{P}_t^p) = z(\mathrm{P}) = \varphi(\bar{\boldsymbol{y}})$: this can be done looking at (Π_t^p) as the maximization of $\varphi(\bar{\boldsymbol{y}}) + g(\boldsymbol{y} - \bar{\boldsymbol{y}})$, where $g()$ is constantly 0 in a non-pointed neighbourhood of $\mathbf{0}$; hence, $\nabla g(\mathbf{0}) = \mathbf{0}$ and the same arguments about the Moreau-Yosida regularization at the beginning of §I.2.2 apply. ♦

Hence, a sequence of NS is aimed at solving (P_t^p), by approximating X (that need not be a polyhedral set) with a sequence of growing polyhedra $\bar{X}_\beta$: note that *(i)* was not used, and it is therefore not needed for this result, while *(iii)* is clearly crucial. The above proof makes use of the compactness of X to prove "primal" convergence, and one might expect that an equivalent "dual" statement could be easily constructed, especially in view of Observation I.2.2.1: surprisingly, this does not appear to be the case. In fact, all the available proofs that can be applied to the $p = 2$ case use a relation [CL93] that does not extend to the linear cases, and the converse also seems to be true: conversely, the present "primal" proof make no distinction at all on the specific norm used. However, it explicitly requires that all the information computed during the course of the algorithm ($\boldsymbol{g}$ or $\boldsymbol{x}$, depending on the viewpoint) be kept until termination: as for Theorem I.2.2.2, one can expect that some milder requirement be sufficient, but the *monotone* β-strategy is not enough to guarantee convergence for $p \in \{1, \infty\}$, due to *nondifferentiability* of the L_1 and L_∞ norm functions ($\boldsymbol{s}_i$ is only a *subgradient*, and therefore not even *strict decrease* can be guaranteed), as the above counterexamples have shown. This motivates a further definition that will not give any finite bound on the dimension of β, but can help to design practical β-strategies.

Definition I.4.2.3: *a β-strategy is **strictly monotone** if it is monotone and, if some $\boldsymbol{g}_j$ has been removed from β after the h-th iteration of the NS sequence, no other removal is permitted unless $v_k - \mu \geq v_i$ ($\mu > 0$), where $k < i$ is the latest iteration where a removal took place.*

A *strictly monotone* β-strategy ensures that $\{ v_i \} \to 0$, preventing removals when $v_i \leq \mu$ (for a large enough i): hence, assuming that a (strictly, if $p \in \{1, \infty\}$) monotone β-strategy is adopted, Theorem I.4.2.2 can be used in place of Theorem I.2.2.2 inside Theorem I.2.3.1, yielding a "weak" convergence proof (with $\varepsilon > 0$) for the new cases. Also, note how the second part of the definition of a *safe* β-strategy is analogous to the definition of a strictly monotone β-strategy applied to the SS rather than to the NS, but for the fact that no guarantee on the decrease has to be given, since $\bar{\varphi}_\beta(\bar{\boldsymbol{y}}_k) - \bar{\varphi}_\beta(\bar{\boldsymbol{y}}_i) = v_k - v_i$ if $\bar{\boldsymbol{y}}$ has not changed in the meantime.

It is now possible to extend Theorem I.2.3.2 (for $\varepsilon = 0$) to the new case: the proof need not be restructured, so that only the differences will be detailed.

Theorem I.4.2.4: *the sequence* $\{ \varphi(\bar{\boldsymbol{y}}_i) \}$ *converges to a maximizer* φ^* *of* $\varphi()$.

Proof: As Theorem I.2.3.2, it is assumed that infinitely many SS are performed, but that there exist a point $\hat{\boldsymbol{y}}$ and a $\delta > 0$ such that $\varphi(\hat{\boldsymbol{y}}) \geq \varphi(\bar{\boldsymbol{y}}_i) + \delta$ for all i: hence, for a sufficiently large i $\hat{\boldsymbol{y}}$ must be outside the ball $B_i = \{ \boldsymbol{y} : \| \boldsymbol{y} - \bar{\boldsymbol{y}}_i \|_q \leq t_i \}$, since now $\bar{\boldsymbol{y}}_{i+1}$ maximizes $\bar{\varphi}_\beta()$ over B_i, so that

$$\varphi(\hat{\boldsymbol{y}}) \geq \varphi(\bar{\boldsymbol{y}}_i) + \delta > \varphi(\bar{\boldsymbol{y}}_i) + v_i = \bar{\varphi}_\beta(\bar{\boldsymbol{y}}_{i+1}) \geq \bar{\varphi}_\beta(\boldsymbol{y}) \geq \varphi(\boldsymbol{y}) \; \forall \, \boldsymbol{y} \in B_i .$$

The relation $v_i \geq \delta (\rho_i / \omega_i)$ is obtained as in the original proof, but in this case the numerator is bounded (since $\rho_i = t_i \geq \underline{t}$), so that

$$\infty > \Sigma_{j = i..\infty} v_j > \delta \Sigma_{j = i..\infty} (t_j / \omega_j) \geq \underline{t} \delta \Sigma_{j = i..\infty} (1 / \omega_j);$$

in order to bound the denominator, we use the *triangle inequality* (see again Figure 12)

$$\| \boldsymbol{w}_{i+1} \|_q \leq \| \boldsymbol{w}_i \|_q + \| -\boldsymbol{d}_i \|_q \leq \| \boldsymbol{w}_i \|_q + t_i$$

that yields $\infty > \Sigma_{j = i..\infty} v_j > \underline{t} \delta \Sigma_{j = i \,..\, \infty} \dfrac{1}{\omega_i + (j - i) \bar{t}}$

i.e. divergent series of the same form, hence the contradiction. ♦

The original proof used the relation $\boldsymbol{d}_i \boldsymbol{w}_i \geq 0$, that is true (for sufficiently large i) only for $p = 2$ because $\boldsymbol{d}_i$ is (proportional to) a (σ_i-)subgradient: this difference is strictly related to the "weakness" of the proofs, since the better theorem for the $p = 2$ case alone is based (to bound the "movements" of $\bar{\boldsymbol{y}}$) on a similar property [CL93] that is *not* true in this more general setting, since $\boldsymbol{d}$ *is not necessarily a* σ*-subgradient of* $\varphi()$ at $\bar{\boldsymbol{y}}$.

Since it seems impossible to identify a convergent subsequence of $\{ \bar{\boldsymbol{y}}_i \}$, and therefore an optimal dual solution $\boldsymbol{y}^*$ (unless we assume *inf-compactness* of $\varphi()$ to ensure that some level set be compact), it is also hard to prove primal optimality of the limit $\boldsymbol{x}^*$ of any converging subsequence of $\{ \bar{\boldsymbol{x}}_i \}$: however, this is not much interesting in practice, because of the even stronger properties that can be proved for polyhedral functions. Furthermore, the proof of Theorem I.2.3.5 (finite convergence in the polyhedral case) is not any better for $p = 2$, i.e. if finite convergence to a true optimum is required, the price of a potentially unlimited memory supply has to be paid, whatever p is: this strongly supports the claim that the choice of p is essentially irrelevant for a Bundle algorithm, at least if applied to polyhedral functions.

Finally, the two alternative algorithmic schemes described at the end of §I.2.2 can clearly be extended to the new case as well: the convergence of each scheme may however require the obvious stronger assumptions on the β-strategy, such as of being strictly monotone instead of just monotone.

I.4.3 Extensions

An important feature of any practical NDO algorithm is the capability of dealing with constraints on

the dual space: Bundle algorithms can be easily extended to cope with constraints $\boldsymbol{y} \in Y$ if Y is (or can be approximated with) a polyhedron $\mathbf{0} \leq \boldsymbol{yH} + \boldsymbol{h}$. Luckily, the majority of constraints encountered in practice are just nonnegativity constraints $\boldsymbol{y} \geq \mathbf{0}$, arising from Lagrangean Relaxation of inequalities (Cf. Sec. II.1.4.2 and §II.5.2), with at most a few general linear constraints, usually involving each only a subset of the variables (Cf. Sec. II.1.4.3 and §II.5.1). The standard approach, in the $p = 2$ case, is to make the direction finding subproblem "aware" of the constraints, that is to solve the modified subproblem

$$(\Pi^2_{\beta t}) \qquad \max_{\boldsymbol{d},v} \left\{ v - 1/2t \left\| \boldsymbol{d} \right\|^2 : v\boldsymbol{e} \leq \boldsymbol{d}\boldsymbol{G}_\beta + \alpha_\beta \ , \mathbf{0} \leq (\bar{\boldsymbol{y}} + \boldsymbol{d})\boldsymbol{H} + \boldsymbol{h} \right\}$$

$$\text{i.e.} \qquad \max_{\boldsymbol{d}} \left\{ \varphi_\beta(\boldsymbol{d}) - 1/2t \left\| \boldsymbol{d} \right\|^2 : (\bar{\boldsymbol{y}} + \boldsymbol{d}) \in Y \right\}$$

or, equivalently, its dual

$$(\Delta^2_{\beta t}) \qquad \min_\theta \left\{ 1/2 \, t \left\| \boldsymbol{G}_\beta\theta + \boldsymbol{H}\mu \right\|^2 + \alpha_\beta\theta + \bar{\boldsymbol{h}}\mu : \theta \in \Theta \ , \mu \geq \mathbf{0} \right\}$$

(where $\bar{\boldsymbol{h}} = \boldsymbol{h} - \bar{\boldsymbol{y}}\boldsymbol{H}$) that ensure feasibility of the trial point $\boldsymbol{y} = \bar{\boldsymbol{y}} + \boldsymbol{d}$: although intuitively clear, the new $(\Pi^2_{\beta t})$ problem can also be justified by a characterisation of the *steepest feasible ascent direction* on a cone, that can be very easily shown to hold in the case of polyhedral functions.

Lemma I.4.3.1: *the steepest* ***feasible*** *ascent direction for* $\varphi_\beta()$ *at* $\bar{\boldsymbol{y}}$ is *the optimal solution of*

$$\max_{\boldsymbol{d},v} \left\{ v - 1/2 \left\| \boldsymbol{d} \right\|^2 : v\boldsymbol{e} \leq \boldsymbol{d}\boldsymbol{G}_\beta \ , \mathbf{0} \leq \boldsymbol{d}\boldsymbol{H} \right\}.$$

Proof: First, note that $\bar{\boldsymbol{h}} = \mathbf{0}$ and $\alpha = \mathbf{0}$, i.e. all the subgradients $\boldsymbol{g}_i$ and constraints $\boldsymbol{h}_i$ that are not "active" in $\bar{\boldsymbol{y}}$ had been discarded: the steepest feasible ascent direction is (Cf. §I.4.1) the solution of

$$\max_{\boldsymbol{d}} \left\{ \min_{\boldsymbol{g}} \left\{ \boldsymbol{d}\boldsymbol{g} : \boldsymbol{g} \in \partial\varphi_\beta(\bar{\boldsymbol{y}}) \right\} : \left\| \boldsymbol{d} \right\|^2 = 1 \ , \mathbf{0} \leq \boldsymbol{d}\boldsymbol{H} \right\}$$

$$\text{i.e.} \qquad \max_{\boldsymbol{d},v} \left\{ v : v\boldsymbol{e} \leq \boldsymbol{d}\boldsymbol{G}_\beta \ , \left\| \boldsymbol{d} \right\|^2 = 1 \ , \mathbf{0} \leq \boldsymbol{d}\boldsymbol{H} \right\}.$$

Then, the required problem is obtained by dualizing the $\left\| \boldsymbol{d} \right\|^2 = 1$ constraint in the latter formulation with multiplier 1: the two are equivalent because the solution does not depend on the Lagrangean multiplier (confront with $(\Delta^2_{\beta t})$: since $\alpha_\beta = \bar{\boldsymbol{h}} = \mathbf{0}$, t has clearly no influence). ♦

The above theorem is only a special case of more general results [LPS96] that calls for the restatement of $(\Pi^2_{\beta t})$ as an unconstrained optimization problem via the *concave characteristic function* (Cf. §I.4.1) of Y and the use of the relations between the *normal cone* to the set of constraints and the subdifferential of their characteristic function (Cf. §II.4.1): however, it already suggests that $(\Pi^2_{\beta t})$ is the steepest *feasible* ascent direction finding problem taking into account "non-local" information, i.e. subgradients and constraints that are not active in $\bar{\boldsymbol{y}}$. The other, obvious interpretation is as just the maximization the "stabilized" CP-model over the feasible set Y, leading by analogy to the definition

$$(\Pi^p_{\beta t}) \qquad \max_{\boldsymbol{d}} \left\{ \varphi_\beta(\boldsymbol{d}) : \mathbf{0} \leq \boldsymbol{d}\boldsymbol{H} + \bar{\boldsymbol{h}} \ , \left\| \boldsymbol{d} \right\|_q \leq t \right\}$$

that, not surprisingly, turns out to have as dual

$$(\Delta^p_{\beta t}) \qquad \min_\theta \left\{ t \left\| \boldsymbol{G}_\beta\theta + \boldsymbol{H}\mu \right\|_p + \alpha_\beta\theta + \bar{\boldsymbol{h}}\mu : \theta \in \Theta \ , \mu \geq \mathbf{0} \right\}$$

for $p \in \{1, \infty\}$ and $q \in \{\infty, 1\}$; the proof is identical to that of the unconstrained case (Cf. §I.4.1). Interestingly enough, this is exactly the subproblem that arises if the primal feasible set X is *not*

compact, i.e. if any point $\boldsymbol{x} \in \bar{X}$ has to be written as $\boldsymbol{x} = \Sigma_j \boldsymbol{x}_j\theta_j + \Sigma_h \boldsymbol{r}_h\mu_h$ ($\theta \in \Theta$, $\mu \geq \boldsymbol{0}$), where $\boldsymbol{x}_j$ are the extreme points and $\boldsymbol{r}_h$ are the *extreme rays* of X: in fact, in this case the Master Problem is

$$(M) \qquad \min_\theta \{ \boldsymbol{c}(\Sigma_j \boldsymbol{x}_j\theta_j + \Sigma_h \boldsymbol{r}_h\mu_h) : \boldsymbol{A}(\Sigma_j \boldsymbol{x}_j\theta_j + \Sigma_h \boldsymbol{r}_h\mu_h) = \boldsymbol{b} , \theta \in \Theta , \mu \geq \boldsymbol{0} \}$$

so that the "constrained" ($\Delta^p_{\beta t}$) is again a Lagrangean relaxation with multiplier t of the "extended" (M) with constraints $\| \boldsymbol{A}(\Sigma_j \boldsymbol{x}_j\theta_j + \Sigma_h \boldsymbol{r}_h\mu_h) - \boldsymbol{b} \|_p = 0$. This is not accidental, as the introduction of dual constraints $\boldsymbol{y} \in Y$ is often dictated by the need of avoiding portions of the dual space where the pricing subproblem (P_{c-yA}) may be unbounded: hence, although it should be possible to extend the "primal" convergence proof of §I.4.2 to this case, the considerable complication of the notations that would be necessary is not worth the result, since in practice the dual set Y is usually easy to find. But for this possible complication, all that has been said about the unconstrained case gracefully extends to the constrained one: note once again the importance of condition *(v)*, which ensures that at least a feasible point be probed during the Line Search.

From the practical viewpoint, the introduction of many constraints in the (QP) subproblems may clearly make it much harder to solve: in fact, it has been shown in §I.3.5 that the BTT algorithm for the box-constrained case requires a possibly long sequence of calls to a method for the unconstrained case. However, nonnegativity constraints $\boldsymbol{y} \geq \boldsymbol{0}$ can still be quite efficiently solved, since usually the "active set" of variables is quite stable: that case is of a large practical relevance, and the relative (QP) will be denoted by ($\Pi^{p\prime}_{\beta t}$) in all Part II.

Another important extension is a different treatment of *separable* functions, i.e.

$$\varphi(\boldsymbol{y}) = \Sigma_{i = 1..k} \varphi^i(\boldsymbol{y})$$

that very often occur in practice: an example with $k = 2$ is the *Lagrangean Decomposition*, where a problem is decomposed by introducing two copies of the same set of variables and then dualizing the equality constraints (Cf. Sec. II.1.4.3), but separable functions with larger k appear in the Lagrangean relaxations of "coupling" constraints in block-structured problems to which Part II is entirely dedicated. In this case, the computation of each $\varphi^i(\bar{\boldsymbol{y}})$ corresponds to an independent minimization problem, that gives a $\boldsymbol{g}^i \in \partial\varphi^i(\bar{\boldsymbol{y}})$: hence, rather than aggregating this information into an unique subgradient $\boldsymbol{g} = \Sigma_{i = 1..k} \boldsymbol{g}^i$, the *disaggregated* problem

$$(\bar{\Pi}^2_{\beta t}) \qquad \max_{\boldsymbol{d},v} \{ \Sigma_{i = 1..k} v^i - 1/2t \| \boldsymbol{d} \|^2 : v^i \leq \boldsymbol{d}\boldsymbol{g}^i_j + \alpha^i_j \;\; j \in \beta^i , i = 1 .. k \}$$

or, equivalently, its dual

$$(\bar{\Delta}^2_{\beta t}) \qquad \min_\theta \{ 1/2\, t \| \Sigma_{i = 1..k} \Sigma_{j \in \beta^i} \boldsymbol{g}^i_j\theta^i_j \|^2 + \Sigma_{i = 1..k} \Sigma_{j \in \beta^i} \alpha^i_j\theta^i_j : \theta^i \in \Theta^i , i = 1 .. k \}$$

can be solved; note that, in the dual, the k separate "Bundles" β^i (one for each $\varphi^i()$) only interacts in the objective function. The approach plainly extends to the $p \in \{1, \infty\}$ cases, as the dual of

$$(\bar{\Pi}^p_{\beta t}) \qquad \max_{\boldsymbol{d},v} \{ \Sigma_{i = 1..k} v^i : v^i \leq \boldsymbol{d}\boldsymbol{g}^i_j + \alpha^i_j \;\; j \in \beta^i , i = 1 .. k , \| \boldsymbol{d} \|_q \leq t \}$$

can be easily shown to be

$$(\bar{\Delta}^p_{\beta t}) \qquad \min_\theta \{ t \| \Sigma_{i = 1..k} \Sigma_{j \in \beta^i} \boldsymbol{g}^i_j\theta^i_j \|_p + \Sigma_{i = 1..k} \Sigma_{j \in \beta^i} \alpha^i_j\theta^i_j : \theta^i \in \Theta^i , i = 1 .. k \}$$

i.e. the usual Lagrangean relaxation of the *disaggregated* Master Problem

$$(\bar{M}) \qquad \min_{\theta} \{ \Sigma_{i=1..k} \boldsymbol{c}^i(\Sigma_{j \in \beta^i} \boldsymbol{x}_j^i \theta_j^i) : \Sigma_{i=1..k} \boldsymbol{A}^i(\Sigma_{j \in \beta^i} \boldsymbol{x}_j^i \theta_j^i) = \boldsymbol{b} \, , \, \theta^i \in \Theta^i \}$$

reformulated in L_p-norm. Use of disaggregate information about $\varphi()$, when available, is widely believed to be beneficial: the sum of the *k disaggregated CP-models* is at least a better description of $\varphi()$ than the aggregate one. Interestingly enough, one of the most convincing proofs of the benefits of disaggregation has been given in [JLF93], where aggregate and disaggregate Dantzig-Wolfe approaches for the solution of Multicommodity Min Cost Flow Problems are compared: there, it is shown that the currently available (LP) solvers are capable of exploiting the sparsity of the *disaggregated vertices* $\boldsymbol{x}_j^i$ in such a way that the reduction in iteration counts is far more important than the increased computational cost of the subproblem.

In both the above extensions, the possibility of using (LP) subproblems rather than (QP) ones is appealing: for the constrained case, for instance, adding the $\boldsymbol{y} \in Y$ constraints seems to be "natural" for $p \in \{1, \infty\}$, while for $p = 2$ a (theoretically) unconstrained maximization problem is turned into a constrained one. Moreover, any "general" (non-box) linear constraint $\boldsymbol{h}_i$ defining Y has the same cost as a subgradient $\boldsymbol{g}_i$ permanently kept in β throughout all the execution of the algorithm, so that (although an active set strategy might be employed to keep the set of constraints small) dealing with "complex" sets Y can be very expensive if using the existing (QP) codes (Cf. §II.5.1 and §II.5.2). Use of the efficient (LP) technology currently available could be instrumental for reducing the cost of the direction finding phase, especially in the constrained and/or separable (with a large k) case: in fact, the size of the (QP) subproblem could easily exceed the capacity of the specialized (QP) codes.

Several other extensions might be useful, but for space reasons can only be sketched here:

- in some applications it is easy to obtain a (tight) upper bound $\bar{\varphi}$ to $z(\Pi^p_{\beta t})$ while calculating $\varphi(\boldsymbol{y})$ (if $\varphi()$ is a Lagrangean function, $\bar{\varphi}$ can be found by heuristics for the original problem): the Bundle code must be able to exploit this information, the best possible way being perhaps that of making the CP model always bounded by adding $\langle \boldsymbol{0} \, , \, \bar{\varphi} - \varphi(\bar{\boldsymbol{y}}) \rangle$ to β, but $\bar{\varphi}$ can be also used to tune several critical parameters of the algorithm such as t^* or $\bar{\varepsilon}$ (Cf. §I.5.3);
- calculating $\varphi(\boldsymbol{y})$ (i.e. solving $(P_{\boldsymbol{c}-\boldsymbol{yA}})$) may be hard, hence it may be necessary to resort to faster inexact algorithms that give approximate solutions: in real applications, it may be impossible even to know any constant estimate of the error, but the function can still be "maximized", in the sense of going towards an hopefully almost optimal solution, if the negative α_i that are created are appropriatedly corrected;
- in some applications, multiple [ε-]subgradient of a given point $\boldsymbol{y}$ may be available, corresponding e.g. to different [ε-]optimal solutions of $(P_{\boldsymbol{c}-\boldsymbol{yA}})$: when the calculation of $\varphi()$ is the most expensive task, it may be sensible to provide the Bundle algorithm with all the possible information.

The above observations call for a more complete description of the implementation issues arising in the design of a practical Bundle code: however, this is avoided here since most of them will be discussed in Part II along with the description of the various Bundle-type codes presented there.

I.5 The *t*-strategies

The practical performances of all the Bundle algorithms are strongly dependent on the on-line choice of one critical *algorithmic parameter* (t, ε and l for ($\Delta^{p}_{\beta t}$), (M^{ε}_{β}) and ($\Lambda_{\beta l}$) respectively) whose value is admittedly hard to guess [HL93 Remark XIV.4.2.1, comments to Algorithm XV.3.3.4]. Most of the *t*-strategies proposed in the literature are essentially heuristics, as described in §I.5.1: however, more sophisticated *t*-strategies have also been devised, like the BT approach to which §I.5.2 is devoted. Unfortunately, all these *t*-strategies share a rather negative behaviour, the "small *t*" phenomenon, at least when used for polyhedral functions with many variables: this is probably due to the fact that all those approaches are "myopic" in the time dimension, and focus exclusively on the "dual" side of the problem (achieving a large value if $\varphi()$) disregarding the "primal" one (finding a null subgradient, or equivalently a feasible primal solution). Hence, in §I.5.3 two "Long-Term" *t*-strategies are proposed that try to avoid these pitfalls, and some encouraging computational results are reported.

I.5.1 Heuristic *t*-strategies

A number of heuristic rules can be proposed as *t*-strategies: surprisingly, only [Ki90] seem to have appeared in the literature. There, the idea is that of approximating the restriction of $\varphi()$ along $\boldsymbol{d}$ with a quadratic function $q()$ such that $q(0) = \varphi(\bar{\boldsymbol{y}})$, $q(t) = \varphi(\boldsymbol{y})$ and having $q'(0) = \boldsymbol{v}$, yielding

(a) $$t_a = \frac{tv}{2(v - \Delta\varphi)};$$

like all the heuristic rules, *(a)* can be questioned in several ways, the basic one being perhaps just wondering why should a quadratic function be any good in predicting the behaviour of a (say) piecewise-linear one. However, even if this is accepted, it is as well possible to rather fix $q'(t) = \boldsymbol{dg}$, yielding the alternative rule

(b) $$t_b = t\frac{2\Delta\varphi - \boldsymbol{dg}}{2(\Delta\varphi - \boldsymbol{dg})} = t\frac{\alpha + \Delta\varphi}{2\alpha}$$

that has the nice property that $t_b > t \Leftrightarrow \boldsymbol{dg} > 0$ (see Figure 21 and Figure 22); another possibility could be to maximize the piecewise linear function constructed by the two linearizations at 0 and $\boldsymbol{y}$ (restricted to $\boldsymbol{d}$), yielding

(c) $$t_c = \frac{\alpha t - 0}{v - \boldsymbol{d}\,\boldsymbol{g}}.$$

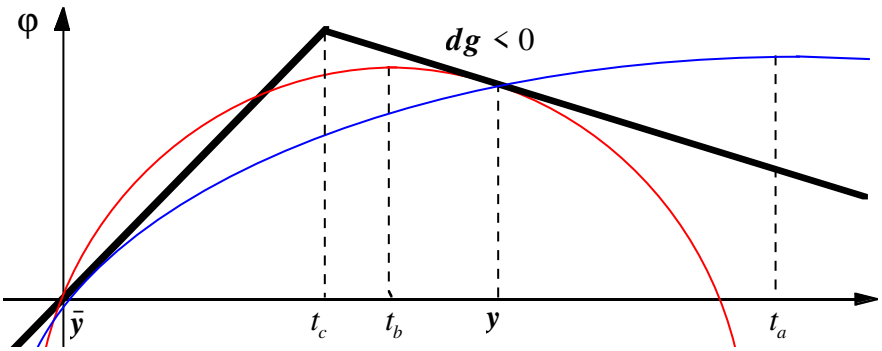


Figure 21: rules *(a)*, *(b)* and (*c*) can give different values of t for $\boldsymbol{dg} < 0$

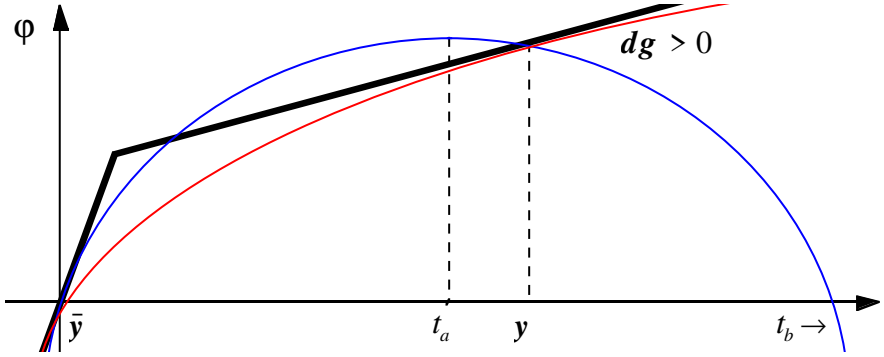


Figure 22: rules *(a)* and *(b)* can give different values of t for $\boldsymbol{dg} > 0$

Each of the above formulas have a "degenerate" case where it is not capable of giving sensible results (respectively $v \approx \Delta\varphi$, $\alpha \approx 0$ and $v \approx \boldsymbol{dg}$), but a more relevant issue of the approach is that only information relative to the current iteration is used: the method can be extended to have some sort of "memory", as in [Ki90] where t is increased or decreased - using *(a)* - only after *three* consecutive SS or NS, but this does not appear to be a really satisfactory solution.

However, the above formulas do *not depend on the choice of p*, i.e. they keep their heuristic validity - if any - in the $p \in \{1, \infty\}$ cases also: surprisingly, this is (almost) true even for the most sophisticated t-strategy proposed so far, the Bundle / Trust-Region method described in the next paragraph.

I.5.2 The Bundle / Trust-Region algorithm

The *Bundle / Trust-Region* (BT) algorithm, introduced in [SZ92], is described in Figure 23: in the pseudo-code, C_1 and C_2 stand for extra conditions that will be discussed later on. The main ideas of the algorithm are standard: in order to accept the new $\boldsymbol{y}$ as the current point, the *obtained* increase $\Delta\varphi$ must be at least a constant fraction (m_1) of the *predicted* increase $v = \varphi_\beta(\boldsymbol{d})$ (condition SS(i)). Measuring *relative* improvements, the SS(i) condition depends on how "demanding" one had been at the previous iteration, i.e. on t, and has a potential drawback in that the *absolute* magnitude of $\Delta\varphi$ is completely disregarded, possibly leading to situations where a negligible improvement is considered important. Condition SS(*ii*) tries to avoid this problem by detecting the case in which a (much) better improvement could have been obtained by augmenting t: since moving to $\boldsymbol{y}$ would nullify the linearization error of newly obtained $\boldsymbol{g}$ ($\in \partial\varphi(\boldsymbol{y})$), the condition $\boldsymbol{dg} + 0 \le m_2 v < v$ ensures that the current $(v, \boldsymbol{d})$ will no more be optimal to ($\Pi_{\beta'}$) if $\boldsymbol{g}$ is added to β, i.e. that the new information has

seriously modified the Cutting Plane model.

Bundle/Trust Region Algorithm

⟨ choose $0 < m_1 < m_2 < 1$, $0 < m_3 < 1$, $t^* > 0$, $\varepsilon > 0$, the initial $\bar{y}$ and t ⟩; $\beta = \{ g(\bar{y}) \}$;
do
⟨ solve $(\Pi^p_{\beta t})$ and $(\Delta^p_{\beta t})$ for (v, d) and $(\sigma, \|z\|_p)$ ⟩;
$y = \bar{y} + d$; $g = g(y)$; $\Delta\varphi = \varphi(y) - \varphi(\bar{y})$; $\alpha = \Delta\varphi - dg$;
if($\Delta\varphi \geq m_1 v$) /* SS(*i*) */
then **if**(($dg \leq m_2 v$) & ~C_1) /* SS(*ii*) */
then $\beta = \beta \cup \{ g \}$; $\bar{y} = y$;
else Increase(t);
else /* NS(*i*) = ~SS(*i*) */
if(($\alpha \leq m_3 \sigma$) & C_2) /* NS(*ii*) */
then $\beta = \beta \cup \{ g \}$;
else Decrease(t);
while($\sigma + t^* \|z\|_p > \varepsilon$);

Figure 23:the Bundle/Trust Region algorithm

However, dg is also the slope of $\varphi_\beta()$ in d, i.e. how much could be gained - at most - by moving to $2d$ instead of d (see Figure 24): hence, ~SS(*ii*) is true if the maximum improvement that might be obtained by doubling t (if d would remain the same *and* $\varphi_\beta()$ would be good approximation of $\varphi()$) is a significative fraction of v; condition C_1 is just "$t \approx \bar{t}$", i.e. the SS has to be done anyway if t is already (almost) as large as it can be. The NS(*ii*) condition is quite clear: the newly obtained subgradient g is trusted as a relevant first-order information about $\varphi()$ in $\bar{y}$ only if it is of about the same "quality" of z, i.e. if it belongs to the $(m_3\sigma)$-subdifferential of $\varphi()$ in $\bar{y}$, m_3 being another fixed constant. Actually, the idea is that m_3 should be smaller than 1, since z is "poor" (a NS has just been done) and therefore something better than z is needed: if it is not available, a t decrease is preferred.

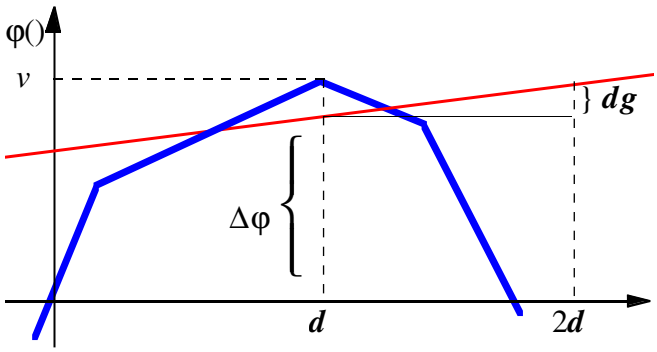


Figure 24: the SS(*ii*) rule and the expected improvement in $\bar{y} + 2d$

However, the interesting point is that the overall algorithm is constructed in such a way that a *Curve-Search* procedure on the y space is performed for seeking the "best" value of t: this is based on the fact that, if SS(*i*) and ~SS(*ii*) are true for a given t, then there exists a $t' > t$ for which both SS(*i*) and SS(*ii*) will be true, and the same holds for the NS() conditions (note that the new g is *not* inserted in β in these cases just for ensuring this property, and that the last part of our condition *(v)* exactly accounts for this extra degree of freedom). Hence, the algorithm maintains a "feasible interval" $[l , u] \subseteq [0 , \bar{t}]$ containing t, and the IncreaseT(), DecreaseT() procedures respectively set $l = t$, $u = t$ and choose the new t in the shrunken interval: some safeguards are necessary, in theory, to guarantee that the interval will eventually become smaller than any fixed threshold in a finite number of steps, and in

fact a simple bisection method is analyzed in [SZ92], but some of the heuristics of §I.5.1 could be used as well. The C_2 condition is just "$l = 0$", telling that the algorithm is performing a Curve Search where SS(i) and ~SS(ii) have been true at least once, so that only a SS will be accepted as its final result (l and u are then reset to $\underline{t}$, $\bar{t}$): in fact, it can be shown that (for $p = 2$) the CS ends in a finite number of steps with either a SS or a NS. Note that this is *not* done to maximize $\varphi()$ - the final SS might well *not* lead to the best point found during the CS - but rather to find a "good" value of t by sampling the space around $\bar{\boldsymbol{y}}$: this is done by letting $\boldsymbol{d}$ change in response to the t variation, although it may be also reasonable to do it along a fixed direction ("resuscitating" the Line Search for a different purpose) if the solution of ($\Delta^p_{\beta t}$) would be the most time-consuming part of the whole approach.

Having been proposed for the $p = 2$ case, the original BT algorithm does not fit exactly in the convergence proofs of Chapter I.2 and §I.4.2: in particular, it does not use a lower bound $\underline{t}$ on t, hence *(iii)* does not necessarily holds, and furthermore the proof that the CS is finite uses the *unicity* of the optimal solution of ($\Pi^2_{\beta t}$), that does not extend to the $p \in \{1, \infty\}$ cases. However, it is easy to see that only minor changes are needed in order to fit it into the new setting: for instance, finite termination of the CS can be also guaranteed by simple conditions like $u - l < \zeta$ (that are in practice advised anyway, and with a not too small $\zeta > 0$). Furthermore, the fact that a SS is always performed at the end of a CS where SS(i) has been true at least once is not needed for convergence, but can still be easily ensured with the simple device of saving the t (and $\boldsymbol{y}$, v ...) of the latest such iteration and restoring it if the CS is stopped at a NS, obtaining an essentially equivalent algorithm. These changes can be easily made in such a way that t is always kept larger than $\underline{t}$, so that (due to the bisection mechanism, or the equivalent safeguards) *(i)*, *(iii)* and *(iv)* are restored in the same time: since SS(i) is just condition *(ii)*, and *(v)* is automatic, BT can be extended to the new class without loosing any of its main characteristics.

I.5.3 Long Term *t*-strategies

In practice, the behaviour of Bundle algorithms on large-scale NDO problems is sometimes not satisfactory: in our experience, all the above t-strategies work well only during an initial phase, while afterwards t rapidly decreases, since the conditions required for a t augment are almost never satisfied. As a consequence, a long sequence of very "short" Serious Steps is performed, resulting in extremely slow convergence: usually, by just keeping t fixed to some "large" value (that, however, must be choosen with a rule of thumb) the algorithms perform much better. Some "philosophical" observations can be made about these difficulties:

- all the above t-strategies lack of any (at least tentative) *long term memory*: only the outcome of the last (few) iteration(s) is used to choose t, so that - even tough the information about $\varphi()$ is kept - Bundle methods suffer from a lack of information about their own past choices;
- all the above t-strategies measure *relative increments*, and the absolute magnitude of the obtainable increments is completely disregarded;
- in all cases, the "ideal" t that is searched for should permit a sequence of SS only, giving a measure of the "curvature" of $\varphi()$, i.e. how "far" it is possible to go before the function "changes too

much": for polyhedral functions arising from Lagrangean relaxations this may not be a wise choice, as they have an enormous number of facets, so that such a t is likely to be that very small;

– finally, none of the above t-strategies takes into account the reduction of $\| z \|_p$, i.e. the "primal side" of the problem, although a small value of the norm is required for termination.

In this paragraph, two new "Long Term" t-strategies are proposed that try to deal with the above problems: from the dual viewpoint, they are essentially based on keeping a *measure* of what a *sensible improvement* is, and using this information for increasing t (or inhibiting the decreases) if necessary; from the primal viewpoint, they can be seen as Dantzig-Wolfe approaches with *repeated but approximated* Phases 0. These strategies - that are *independent* from the choice of p - are not designed to *replace* the above (heuristic or not) t-strategies, but rather to *integrate* them: they should be applicable to almost every known Bundle algorithm, such as BT of §I.5.2, with only minor modifications in its original rules.

The "Hard" version, described in Figure 25, keeps an "expected minimum increase" $\bar{\varepsilon}$, which is the "Long Term memory" of the method: a step to $\boldsymbol{y} = \bar{\boldsymbol{y}} + \boldsymbol{d}$ is considered useless if the maximum obtainable improvement, measured by v, is not large enough, i.e. greater than a constant fraction $m_4 \leq 1$ of $\bar{\varepsilon}$. In this case, enlarging t is preferred: the new t is chosen by using *sensitivity analysis* information about the $(\Delta^p_{\beta t})$ subproblem, i.e. the two numbers $s \geq 0$ and q (Cf. §I.3.6 and §I.1.2 for $p = 2$) such that $ts + q$ is an upper bound on $v(t)$, and it is set in such a way that $v \geq \bar{\varepsilon}$ at the next iteration. Since $ts + q$ is only an upper bound on $v(t)$, more than one US (but only finitely many) may be required to find the correct value of t.

Hard Long Term t-strategy

⟨ choose $t^* > 0$, $\bar{\varepsilon} \geq \varepsilon > 0$, the initial $\bar{\boldsymbol{y}}$, t and β ⟩;
do
  ⟨ solve $(\Pi^p_{\beta t})$ and $(\Delta^p_{\beta t})$ for $(v, \boldsymbol{d})$ and $(\sigma, \| z \|_p)$ ⟩;
  if($\sigma + t^* \| z \|_p \leq \bar{\varepsilon}$) /* convergence for $\bar{\varepsilon}$ */
    $\bar{\varepsilon} = (\sigma + t^* \| z \|_p) / 2$;
  if(($v < m_4 \bar{\varepsilon}$) & ($t < t^*$)) /* Useless Step */
    $t = \min\{ t^* , (\bar{\varepsilon} - q) / s \}$;
  else /* a "normal" step */
    ⟨ any Bundle algorithm, eventually modified to
      "avoid interference" with the US mechanism ⟩;
while($\sigma + t^* \| z \|_p > \varepsilon$);

Figure 25: the "hard" Long Term t-strategy

$\bar{\varepsilon}$ is decreased once no other increments of the current magnitude are expected, i.e. when *an $\bar{\varepsilon}$-optimal solution is obtained* (assuming that t^* has been correctly chosen): note that $s = 0$ ($\Rightarrow (\bar{\varepsilon} - q) / s = \infty$) implies that the CP model is bounded and the current $\bar{\boldsymbol{y}}$ is q-optimal, so that setting $t = t^*$ will force "convergence" at the next step. In this case, our new stopping rule is clearly benefical, since the old "$\| z \|^2 \leq \varepsilon_Q$ & $\sigma \leq \varepsilon_L$" one would have required two different $\bar{\varepsilon}_Q$ and $\bar{\varepsilon}_L$; needless to say, this approach extends to the use of a *relative* $\bar{\varepsilon}$ as outlined at the end of §I.1.1. Note that s is smaller than the largest

directional derivative of $\varphi_\beta()$ in a properly defined ball B (of "radius" $\bar{t}$) around $\bar{\boldsymbol{y}}$, hence $s \le L$ for some Lipschitz constant $L > 0$: therefore, a lower bound exists on the t increment of a US, since $v = ts + q < m_4\bar{\varepsilon}$ and $t' = (\bar{\varepsilon} - q) / s$, so that

$$t' - t > (\bar{\varepsilon} - q) / s - (m_4\bar{\varepsilon} - q) / s = (1 - m_4)(\bar{\varepsilon} / s) \ge (1 - m_4)(\bar{\varepsilon} / L).$$

When $v \ge m_4\bar{\varepsilon}$, a "normal" Bundle algorithm is used, possibly modified for ensuring that conditions *(i)* - *(v)* are respected: usually, only minor changes are needed, as testified by the case of the BT approach, that has the most sophisticated t-strategy in the literature. For the purpose of the convergence proof, an iteration comprises a "normal" step and all the following consecutive USs (that are nothing but a t-strategy): it is easy to see that the only critical point is the analysis of an infinite sequence of NS, since conditions *(i)*, *(ii)*, *(iv)* and *(v)* are clearly not harmed. The US mechanism does not interfere with a CS started by an SS(*i*) condition: in fact, no US can be performed after an iteration h where SS(*i*) holds, until the CS ends, since β is not changed, $v_h \ge m_4\bar{\varepsilon}$ and $t \ge t_h$ due to the bisection mechanism, that is reset only after the SS. Furthermore, the standard BT scheme can be easily modified in such a way that *after an US,* NS(*i*) *and* ~NS(*ii*) *does not start a CS until* $\bar{\varepsilon}$ *is decreased*: t decreases are *not* totally inhibited, since a CS is still started if SS(*i*) and ~SS(*ii*) hold, but any such CS ends in a SS, so that within any infinite sequence of NS t is never decreased. For *(iii)* to hold, t must also be increased only finitely many times: however, this is clearly true since t is upper bounded and it increases of at least a fixed quantity at each US (since $\bar{\boldsymbol{y}}$ is not changing).

This t-strategy is a refined version of the "dual/primal" scheme at the end of §I.2.3, where convergence is reached for the "real" ε, t is all of a sudden increased to t^* and a sequence of NS is started: from the primal viewpoint, a Bundle algorithm using the "Hard" t-strategy is a DW approach with *partial and repeated* Phases 0, as convergence with a given $\bar{\varepsilon}$ requires $\| \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}} \|_p \le \bar{\varepsilon} / t^*$. In fact, in practice this t-strategy is "inactive" as long as "good" improvements are obtained, and then a sequence of NS + US is performed until the "convergence" is attained: the "t increasing phase" can occasionally be interrupted by a SS, but it usually continues immediately after. The computational results presented below show that this t-strategy can help in correctly sizing t, avoiding the "short steps" phenomenon: however, as $\bar{\varepsilon}$ decreases the "t increasing phases" tend to be longer and longer, and during such phases only a little progress (in terms of $\varphi()$ value) is made towards the optimum.

Although remarkably better than no Long Term strategy at all, the "Hard" t-strategy still seems to be ameliorable: in view of the above observations, a "Soft" approach is proposed in Figure 26 that, rather than forcing t to augment, just inhibits t decreases whenever v is "too small".

Soft Long Term *t*-strategy

⟨ choose $t^* > 0$, $\varepsilon > 0$, the initial $\bar{\boldsymbol{y}}$, t and β ⟩; $\bar{\varepsilon} = \infty$;
do
  ⟨ solve $(\Pi^p_{\beta t})$ and $(\Delta^p_{\beta t})$ for $(v, \boldsymbol{d})$ and $(\sigma, \| z \|_p)$ ⟩;
  $\bar{\varepsilon} = \min(\bar{\varepsilon} , (\sigma + t^*\| z \|_p) / 2)$;
  ⟨ any Bundle algorithm, where t decreases are "substantially" inhibited if $v < \bar{\varepsilon}$ ⟩;
while($\sigma + t^*\| z \|_p > \varepsilon$);

Figure 26: the "soft" Long Term t-strategy

Still, the exact details shall be dependent on the "basic" Bundle algorithm: for BT, it is only necessary to change the NS(*ii*) condition into "($\alpha \le m_3\sigma \vee v \ge \bar{\varepsilon}$) & $l = \underline{t}$", allowing t to decrease during any CS where the SS(*i*) condition has been true in at least one iteration, but avoiding t decreases if v is "small" and a t increase has not been performed previously; since *(iii)* is guaranteed by the standard BT algorithm, there is in fact nothing to do to prove convergence of this variant.

Both these *t*-strategies have some possible pitfalls: for instance, they probably depend onto a "good" choice of t^* for being effective in practice; furthermore, the Hard *t*-strategy is also dependent on the initial choice of $\bar{\varepsilon}$, and the computational experience suggests that the US mechanism should not come into play too early, a reasonable choice for $\bar{\varepsilon}$ being some small multiple of ε. However, t^* has to be guessed only within an order of magnitude, and, being in the same "measure units" of t, $\underline{t}$ and $\bar{t}$, it is usually easy to guess by looking at the t values in some trial runs; furthermore, it seems to depend only on the "scaling" of the function, i.e. to be quite constant on instances of the same class.

The following computational results display the effect of the Long Term strategies in the context of a real-word application to the Reserve Problem in telecommunications of §II.5.2: the "basic" Bundle algorithm uses $p = 2$ and heuristic *t*-strategies based on the *(a)* and *(b)* formulas, but it has shown to be computationally equivalent to BT. Table 5 reports the results for 9 instances, where v is the number of Lagrangean variables and *iter*, *time* are respectively the number of ($\Delta^p_{\beta t}$) solved and the total time taken by the algorithm to converge (with a relative $\varepsilon = 10^{-4}$): each problem has been solved with no Long Term *t*-strategy (*N*), with the Soft one (*S*) and with the Hard one (*H*), but for all the rest the code and the algorithmic parameters (m_1, m_3, t^* ...) were exactly the same. In all but one problem, the algorithm without any Long Term strategy was not able to converge within the prescribed accuracy in the maximum fixed number of iterations (1000 for the 6 smaller instances, 2000 for the 3 larger ones): for all the problems, (S) required less iterations than (H), but when using (H) the number of $\varphi()$ evaluations may be smaller than the number of ($\Delta^p_{\beta t}$) solved, and in fact the running time of (H) is sometimes smaller than that of (S).

	N		S		H	
v	*iter*	*time*	*iter*	*time*	*iter*	*time*
783	1000	38	294	22	471	28
812	1000	83	457	36	443	25
783	1000	34	311	21	595	26
825	1000	90	761	77	796	51
644	1000	30	434	24	950	47
810	1000	23	270	15	480	16
3844	2000	1230	520	412	660	428
3844	861	615	257	314	355	254
3844	2000	774	443	352	788	681

Table 5: results for 9 real-world problems

A picture of the three variants is given in Chart 7 and Chart 8, where the { $\varphi(\bar{y}_i)$ } and { t_i } sequences (respectively) are compared for the three methods when applied to the last instance: since the variants behave almost identically for the first iterations, where they obtain a large part of the total $\varphi()$ increase, the charts start from iteration 30 in order to make the "long term" behaviour more apparent. Comparing the charts, it is easy to see that a larger value of t usually corresponds to a faster convergence; on the other hand, during the "t increasing sequences" of the Hard *t*-strategy (clearly visible in Chart 8, especially the second one that reaches "convergence" near to iteration 300) the curve of Chart 7 is rather "flat", i.e. there is little (if any) progress towards the optimum. The heuristic *t*-strategies alone are not able to avoid t becoming very small (here, $\underline{t} = 10^{-3}$), so that the algorithm essentially ceases to obtain any significative improvement, although it "believes" to be doing well: on

the contrary, both the Hard and the Soft Long Term strategies are capable of keeping t large enough. For these instances, the Soft t-strategy seems to be preferable to the Hard one: heuristically, this can be justified by the fact that the "long term memory" $\bar{\varepsilon}$ in Hard t-strategy is set in a much more "exact" way than in the Soft one, thus requiring more resources that could be better used otherwise.

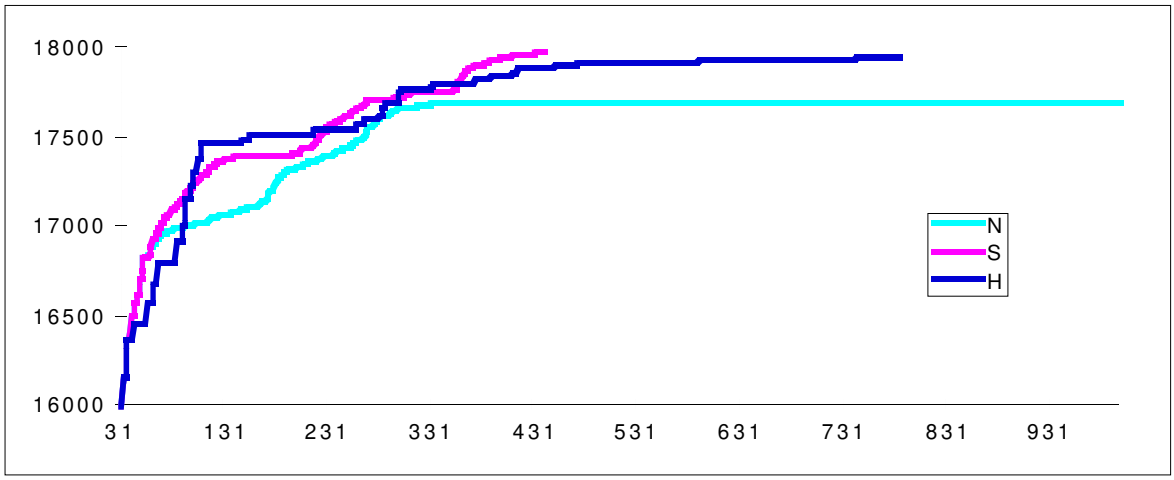


Chart 7: evolution of the value of $\varphi(\bar{\mathbf{y}}_i)$ for the three Long Term t-strategies

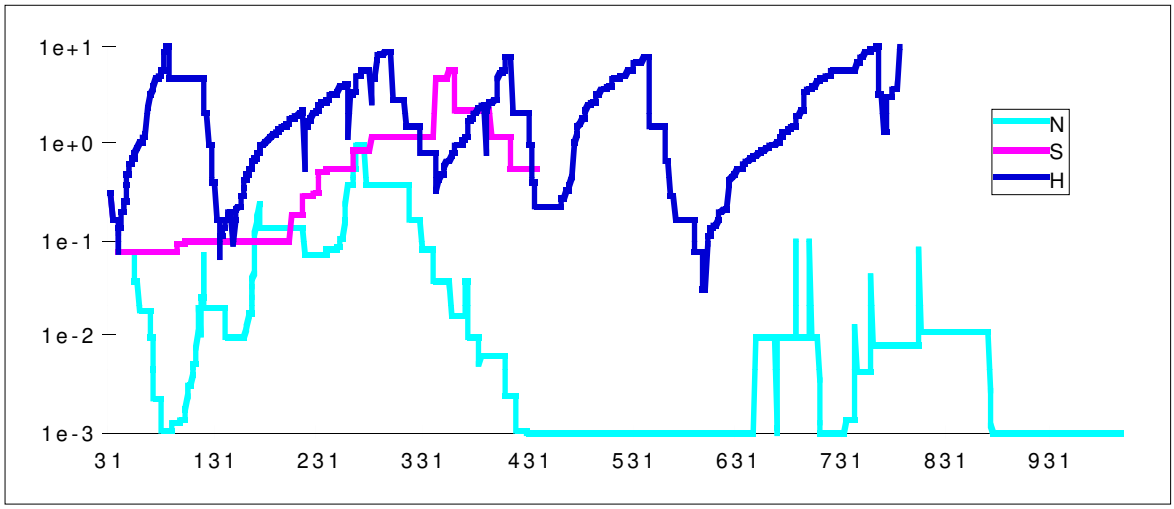


Chart 8: evolution of the value of t_i for the three Long Term t-strategies

From these results, the two Long Term strategies seem to be promising: the choice between the two - or further variants - should be dictated by the characteristics of the problem to be solved, typically the relative cost of the $\varphi()$ calculation and the $(\Delta^p_{\beta t})$ solution. Still, none of the above t-strategies is close to being "optimal", i.e. achieving a "stable" speed of convergence: however, they seem to obtain a significant improvement on the heuristic rules that have been used so far, hence there is a reasonable hope that further advances will be made in the future.

All the t-strategies presented in this Chapter have been initially proposed for the $p = 2$ case, but can be applied to the $p \in \{1, \infty\}$ cases as well: if, as it appears likely, their effectiveness is conserved in the new context, the way is open to the development of codes that exploit the efficient (LP) software currently available to diminish the cost of the "core" subproblem solution, while retaining the good behaviour of the current "quadratic" Bundle implementations.