In-Out Separation and Column Generation Stabilization by Dual Price Smoothing

Artur Alves Pessoa, Ruslan Sadykov, Eduardo Uchoa, François Vanderbeck

In-Out Separation and Column Generation Stabilization by Dual Price Smoothing

A. Pessoa (1), R. Sadykov (2,3), E. Uchoa (1), and F. Vanderbeck (3,2)

(1) LOGIS , Universidade Federal Fluminense, Brazil
(2) INRIA Bordeaux, team RealOpt, France
(3) University of Bordeaux, Institut of Mathematics, France
INRIA research report hal-00750412, March 2013.

Abstract. Stabilization procedures for column generation can be viewed as cutting plane strategies in the dual. Exploiting the link between in-out separation strategies and dual price smoothing techniques for column generation, we derive a generic bound convergence property for algorithms using a smoothing feature. Such property adds to existing in-out asymptotic convergence results. Beyond theoretically convergence, we describe a proposal for effective finite convergence in practice and we develop a smoothing auto-regulating strategy that makes the need for parameter tuning obsolete. These contributions turn stabilization by smoothing into a general purpose practical scheme that can be used into a generic column generation procedure. We conclude the paper by showing that the approach can be combined with an ascent method, leading to improved performances. Such combination might inspire novel cut separation strategies.

Keywords: Column Generation, Stabilization, Cutting Plane Separation

Introduction

Separation strategies from the cut generation literature and algorithmic strategies for stabilization in column generation algorithms are dual counterparts. The pricing procedure in column generation is understood as a separation routine for the master dual. Therefore, efficient strategies to define the separation point or select cuts translate into stabilization techniques and column generation strategies, as emphasized in several papers including [2, 3, 10, 11]. In this paper, we specifically formalize the link between in-out separation [2, 5] and dual price smoothing techniques whereby the price vector used for column generation is defined as a combination of the optimal solution over the current polyhedral approximation of the master dual (denoted π^{out} hereafter) and a feasible dual solution for the true master (denoted π^{in}). We show that dual price smoothing schemes (such as that of [8, 12]) can be understood as an extension of in-out separation, introducing an in-point updating strategy that relies on a valid dual bound computation. Note that dual price smoothing addresses at once the dual oscillations, tailing-off, and degeneracy drawbacks of the column generation procedure. It acts through both smoothing and centralization, and it is simple to implement. Our work brings an additional quality to smoothing. Our proposal for a parameter self-adjusting scheme allows one to avoid the drawback of many alternative stabilization approaches (such as the popular piecewise linear penalty functions [4]) that require fine tuning of several parameters.

More specifically, the contributions of the paper are:

- Establishing the detailed properties of a generic smoothing scheme (that encompasses several variants), including a bound convergence property that has no equivalent in a general in-out procedure. For already existing results, the link with in-out separation has lead to simpler proofs under weaker conditions.
- Proposing a simple scheme for dealing in practice with a sequence of *mis-pricings* (a mis-pricing is a failure to separate π^{out}) that impairs convergence.
- Developing a parameter self-adjusting scheme for automatic tuning that uses gradient information. The scheme is shown to experimentally reproduce the best results obtained by fine tuning the single but critical parameter of the smoothing procedure, essentially making the method parameter-tuning-free. We emphasize that the performance of smoothing techniques, and more generally stabilization techniques, highly depends on proper parameter tuning that is moreover instance dependent. Hence, our automated scheme has practical significance, transforming smoothing into a general purpose technique well suited for a generic branch-and-price solver.
- Extending the smoothing paradigm by combining it with an ascent method that is experimentaly shown to lead to significant improvements.

The paper places dual price smoothing as a key technique in the context of existing column generation stabilization strategies. The features that we introduced in smoothing techniques could inspire dual strategies for cutting plane separation.

1. Column Generation

Below we review the main concepts underlying column generation approaches in order to emphasize the properties on which smoothing schemes rely. Consider the integer program:

$$[\text{F}] \equiv \min\{cx : x \in X\} \tag{1}$$

where

$$X := Y \cap Z \text{ with } Y := \{x \in I\!R^n_+ : \ Ax \geq a\} \text{ , and } Z := \{x \in I\!N^n : \ Bx \geq b, l \leq x \leq u\}.$$

In the decomposition of system X, it is assumed that Z defines a "tractable" subproblem (assumed to be non-empty and bounded to simplify the presentation), but $Ax \geq a$ are "complicating constraints". In other words, we assume that subproblem

$$[\text{SP}] \equiv \min\{cx : x \in Z\} \tag{2}$$

is "relatively easy" to solve compared to problem [F]. Then, a natural approach to solve [F], or to estimate its optimal value, is to exploit our ability to optimize over Z. We review this technique below.

Let Q be the enumerated set of subproblem solutions (it is a finite set given the boundedness of Z), i.e. $Q = \{z^1, \ldots, z^{|Q|}\}$ where $z^q \in Z$ is a subproblem solution vector. Abusing notations, $q \in Q$ is used hereafter as a short-cut for $z^q \in Q$. Thus, we can reformulate Z and *conv*(Z), the convex-hull of the integer solution to Z, as:

$$Z = \{x \in I\!R^n_+ : x = \sum_{q \in Q} z^q \lambda_q, \ \sum_{q \in Q} \lambda_q = 1; \ \lambda_q \in \{0,1\} \ \forall q \in Q\} \ , \tag{3}$$

$$\textit{conv}(Z) = \{x \in I\!R^n_+ : x = \sum_{q \in Q} z^q \lambda_q, \ \sum_{q \in Q} \lambda_q = 1, \ \lambda_q \geq 0 \ \forall q \in Q\} \ . \tag{4}$$

Note that *conv*(Z) defines an ideal formulation for Z. Hence, [SP] can be rewritten as:

$$\text{[SP]} \equiv \min\{cx : x \in Z\} \equiv \min\{cz^q : q \in Q\} \equiv \min\{cx : x \in \mathit{conv}(Z)\}. \tag{5}$$

Exploiting the assumption that the subproblem is tractable, one can derive dual bounds for the original problem [F] by Lagrangian relaxation of the constraints $Ax \geq a$. For any Lagrangian penalty vector $\pi \in I\!R^m_+$, the *Lagrangian function*,

$$L(\pi, x) := \pi\, a + (c - \pi A)x\ , \tag{6}$$

is optimized over Z to yield a valid dual bound on [F], by solving the *Lagrangian subproblem*:

$$\text{[LSP}(\pi)\text{]} \equiv L(\pi) := \min_{x \in Z} L(\pi, x)\ . \tag{7}$$

The *Lagrangian dual function* is defined by $L : \pi \in I\!R^m_+ \rightarrow L(\pi)$. Maximizing function L leads to the best dual bound that can be derived from the Lagrangian relaxation. The *Lagrangian dual problem* is defined as:

$$\text{[LD]} \equiv \max_{\pi \in I\!R^m_+} L(\pi)\ . \tag{8}$$

The Lagrangian dual problem can be reformulated as a max-min problem, or as a linear program:

$$\begin{aligned}
\text{[LD]} &\equiv \max_{\pi \in I\!R^m_+} \min_{x \in Z}\{\pi\, a + (c - \pi A)x\}; && (9)\\
&\equiv \max\{\eta, && (10)\\
&\quad \eta \leq cz^q + \pi(a - Az^q) \quad \forall q \in Q, && (11)\\
&\quad \pi \in I\!R^m_+, \eta \in I\!R^1\}; && (12)\\
&\equiv \min\{\sum_{q \in Q} (c\, z^q)\lambda_q, && (13)\\
&\quad \sum_{q \in Q} (Az^q)\lambda_q \geq a, && (14)\\
&\quad \sum_{q \in Q} \lambda_q = 1, \quad \lambda_q \geq 0 \quad \forall q \in Q\}; && (15)\\
&\equiv \min\{cx : Ax \geq a,\ x \in \mathit{conv}(Z)\ \}. && (16)
\end{aligned}$$

The *Dantzig-Wolfe reformulation* is a valid reformulation of [F] expressed in terms of variables λ_q that were introduced for the reformulation of Z given in (3). Its linear programming (LP) relaxation, which we denote by [M], is precisely the form (13-15) of [LD].

A *column generation* procedure to solve [M] proceeds as follows. At a stage t, the restriction of [M] to columns defined from $Q^t = \{z^1, \ldots, z^t\}$ is denoted by [M^t]. This *restricted master LP* is:

$$\text{[M}^t\text{]} \equiv \min\{\sum_{\tau=1}^{t} c\, z^\tau \lambda_\tau : \sum_{\tau=1}^{t} Az^\tau \lambda_\tau \geq a; \sum_{\tau=1}^{t} \lambda_\tau = 1; \lambda_\tau \geq 0,\ \tau = 1, \ldots, t\} \tag{17}$$

Linear program [M^t] is solved to optimality. Let λ^t denote an optimal solution to [M^t]. Its projection in X is:

$$x^t := \sum_{\tau=1}^{t} z^\tau\ \lambda^t_\tau\ . \tag{18}$$

Let $c\,x^t$ denote its objective value. The linear program dual of [M^t] is:

$$[\text{DM}^t] \equiv \max\{\eta : \pi(Az^\tau - a) + \eta \leq cz^\tau,\ \tau = 1, \ldots, t; \pi \in I\!R^m_+; \eta \in I\!R^1\} \tag{19}$$

Let (π^t, η^t) denote an optimal solution to [DM^t]. Using this dual solution, one searches for the most negative reduced cost column, by solving the subproblem:

$$z^{t+1} \leftarrow z_{\pi^t} := \mathit{argmin}_{x \in Z}\{(c - \pi^t A)x\}\ . \tag{20}$$

If $(c - \pi^t A)\ z_{\pi^t} + \pi^t\ a - \eta^t < 0$, then z_{π^t} defines a negative reduced cost column that is added to the restricted master. Otherwise, the current LP solution is optimal for the unrestricted master program [M].

The above algorithm outputs a sequence of values for the Lagrangian price vector: $\{\pi^t\}_t$, that converges towards an optimal dual price vector, π^*. In the process, one can also derive a sequence of candidate primal solutions, $\{x^t\}_t$, converging towards an optimal solution x^* of problem (16). One can observe the following properties:

Observation 1
(i) *The vector x^t defined in* (18) *is a solution to [M^t]* $\equiv \min\{cx : Ax \geq a,\ x \in \text{conv}(\{z^1, \ldots, z^t\})\}$.
(ii) *The dual solution of [DM^t] is such that* $\pi^t = \text{argmax}_{\pi \in I\!R^m_+} L^t(\pi)$ *where $L^t()$ defines an approximation of the Lagrangian dual function $L()$, considering only the subset of subproblem solutions $\{z^1, \ldots, z^t\}$: i.e.,*

$$L^t() : \pi \rightarrow \min_{z \in \{z^1, \ldots, z^t\}} \{\pi a + (c - \pi A)z\} = \min\{L(z^1, \pi) \ldots, L(z^t, \pi)\}\ . \tag{21}$$

Function $L^t()$ is an upper approximation of function $L()$: $L^t(\pi) \geq L(\pi)\ \forall \pi \in I\!R^m_+$. The hypograph of function $L^t()$ defines a polyhedral outer approximation (illustrated in the left of Figure 1) of the master LP dual program (10-12). By duality, $L^t(\pi^t) = \eta^t = c\,x^t$.
(iii) *Solving [LSP(π^t)] exactly serves four purposes simultaneously:*
$(iii.a)$ *it yields the most negative reduced cost column: $z^{t+1} = z_{\pi^t} \in Q \setminus Q^t$ for [M];*
$(iii.b)$ *it yields the most violated constraint defined by a subproblem solution $z^q \in Q \setminus Q^t$ for [DM];*
$(iii.c)$ *the constraint violation of the oracle solution z_{π^t} defines a sub-gradient of $L(.)$ at point π^t:*

$$g^t := (a - A\ z_{\pi^t})\ ; \tag{22}$$

$(iii.d)$ *the correct value of the Lagrangian function $L()$ is now known at point π^t: $L(\pi^t) = \pi^t a + (c - \pi^t A) z_{\pi^t}$, and therefore this value remains unchanged in any further approximation of $L()$, i.e., $L^\tau(\pi^t) = L(\pi^t)\ \forall \tau > t$.*
(iv) *At stage t,* $\text{conv}(\{(\pi^\tau, L^{\tau+1}(\pi^\tau))\}_{\tau=1,\ldots,t})$ *defines an inner approximation (illustrated in the right of Figure 1) of the master LP dual program (10-12). Outer and inner approximation are equal at these points as $L^{\tau+1}(\pi^\tau) = L(\pi^\tau)$. One of these points defines the incumbent dual solution $\hat{\pi} = \text{argmax}_{\tau=1,\ldots,t} L^{\tau+1}(\pi^\tau)$ with value $\hat{L} = L(\hat{\pi}) = \hat{\eta}$.*
(v) *If $L^t(\pi^t) = \hat{L}$, or equivalently $cx^t = \hat{L}$, then the optimal solution is reached, i.e., $\eta^* = \hat{L}$.*

In the sequel, (π^*, η^*) denotes an optimal solution to the Lagrangian dual, while $\hat{L} = L(\hat{\pi})$ denotes the current best dual (lower) bound on η^*.

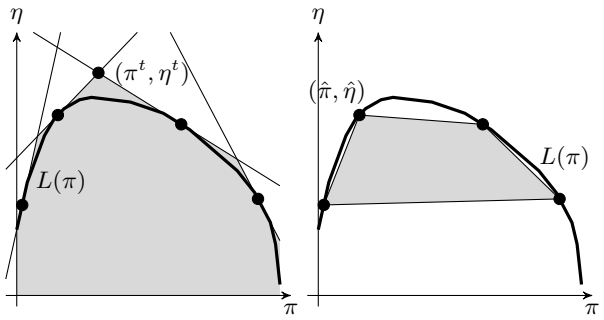


Fig. 1: Outer and Inner polyhedral approximation for the dual master polyhedron [DM] (10-12).

2. Stabilization Techniques in Column Generation

The above column generation procedure, also known as Kelley's cutting plane algorithm for the dual master, yields a sequence of dual solution candidates $\{\pi^t\}_t$ converging towards optimal prices, π^*. The sequence of primal solution candidates $\{x^t\}_t$ is a by-product used to prove optimality of the dual solution. Stabilization techniques are devised to accelerate the convergence of the dual sequence $\{\pi^t\}_t$ towards π^* by targetting the following drawbacks, as listed in [11]:

- *Dual oscillations:* Solutions π^t jump erratically. One extreme solution of the restricted dual master (10-12) at iteration t, [DM^t], is followed by a different extreme point of [DM^{t+1}], leading to a behavior often refered to as *"bang-bang"*. Because of these oscillations, it might be that $||\pi^{t+1} - \pi^*|| > ||\pi^t - \pi^*||$. Moreover, the dual bounds $L(\pi^t)$ are converging non monotically, with ups and downs in the value curve (the *yo-yo* phenomenon).
- *The tailing-off effect:* Towards the end of the algorithm, added inequalities in [DM^t] tend to cut only a marginal volume of the dual solution space, making progress very slow.
- *Primal degeneracy and alternative dual optimal solutions:* An extreme point λ of polyhedron [M^t] has typically fewer non zero values than the number of master constraints. The complementary dual solution solves a system with fewer constraints than variables that admits many alternative solutions. As a consequence, the method iterates between alternative dual solutions without making any progress on the objective value.

Techniques to stabilize column generation belongs to one of the three standard families listed in [11]:

Penalty functions: A penalty is added to the dual objective function to drive the optimization towards dual solutions that are close to a *stability center*, typically defined as the incumbent dual solution $\hat{\pi}$. The dual problem (19) is replaced by

$$\pi^t := \mathit{argmax}_{\pi \in I\!R^m_+} \{L^t(\pi) - \hat{S}(\pi)\}, \tag{23}$$

where the *penalty function*,

$$\hat{S} : \pi \in I\!R^m_+ \rightarrow I\!R_+ \ ,$$

is typically convex, takes value zero at $\hat{\pi}$, and increases as $||\pi - \hat{\pi}||$ increases. The Bundle method [3] is a special case where $\hat{S}(\pi) = \frac{1}{2\theta}||\pi - \hat{\pi}||^2$. One can also make use of a *piecewise linear penalty function* S (see [4] for instance) in order to ensure that the master problem is still a linear program (with additional artificial

variables whose costs and bounds are chosen to model a piecewise linear stabilizing function). Penalty function methods require delicate tuning of several parameters.

Smoothing techniques: The dual solution π^t used for pricing is "corrected" based on previous dual solutions. In particular, Neame [8] proposes to define smoothed price as:

$$\tilde{\pi}^t = \alpha\tilde{\pi}^{t-1} + (1-\alpha)\pi^t \ , \tag{24}$$

i.e., $\tilde{\pi}^t$ is a weighted sum of previous iterates: $\tilde{\pi}^t = \sum_{\tau=0}^{t}(1-\alpha)\alpha^{t-\tau}\pi^\tau$. Wentges [12] proposes another smoothing rule where:

$$\tilde{\pi}^t = \alpha\hat{\pi} + (1-\alpha)\pi^t \ . \tag{25}$$

i.e., $\tilde{\pi}^t = \hat{\pi} + (1-\alpha)(\pi^t - \hat{\pi})$, which amounts to taking a step of size $(1-\alpha)$ from $\hat{\pi}$ in the direction of π^t. In both rules, $\alpha \in [0,1)$ parameterizes the level of smoothing. The pricing problem is then solved using the smoothed prices, $\tilde{\pi}^t$, instead of π^t:

$$z_{\tilde{\pi}^t} := \mathit{argmin}_{x\in Z}\{(c - \tilde{\pi}^t A)x\} \ . \tag{26}$$

Solving this modified pricing problem might not yield a negative reduced cost column, even when one exists for π^t. This situation is the result of a *mis-pricing*. In such case, applying (24) or (25) with the same π^t solution leads to a new dual price vector that is closer to π^t. Note moreover that the incumbent $\hat{\pi}$ is updated each time the current Lagrangian bound improves over $\hat{L}$.

Centralized prizes: One makes faster progress in improving the polyhedral outer approximation of the master LP dual program (10-12) when separating a point (π, η_π) in the interior of (10-12) rather than an extreme point. The analytic-center cutting-plane method (ACCPM) [6] defines iterate π^t as the analytic center of the linear program (10-12) augmented with an optimality cut $\eta \geq \hat{L}$ that defines a trust region. Alternatives exist to keep a formulation of the master as a linear program (see [7] for instance and the references in [11]).

Note that using a smoothed price vector or an interior point for pricing has a drawback. The pricing problem can be harder for some solvers, as there are typically fewer non zero components and less clear dominance that can be exploited in dynamic programming recursions for instance.

3. The link with In-Out Separation

The above smoothing techniques are related to the in-out separation scheme of [2, 5]. The solution over the outer approximation of dual polyhedron (10-12) at iteration t (see Figure 1) defines an *out-point*, i.e., a point outside polyhedron (10-12):

$$(\pi^{\text{out}}, \eta^{\text{out}}) := (\pi^t, L^t(\pi^t)) \ . \tag{27}$$

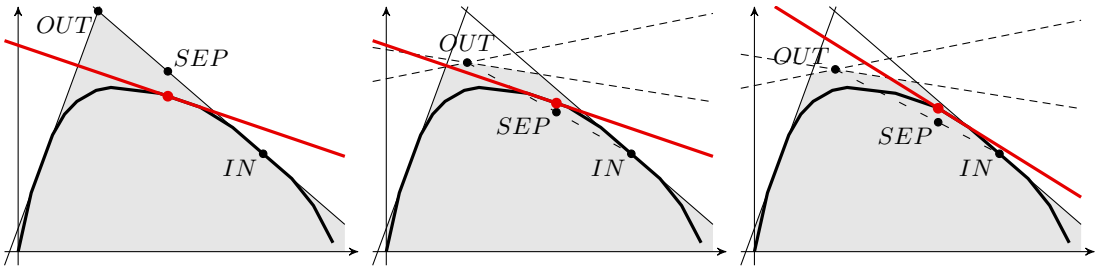
Symetrically, consider a point inside the inner approximation of polyhedron (10-12). Possible definitions of such *in-point* are provided by the smoothing rules described above:

$$(\pi^{\text{in}}, \eta^{\text{in}}) := \begin{cases} (\tilde{\pi}^{t-1}, L(\tilde{\pi}^{t-1})) & \text{under rule (24),} \\ (\hat{\pi}, \hat{L}) & \text{under rule (25).} \end{cases} \tag{28}$$

These are in-points, because $L(\tilde{\pi}^{t-1})$ and $L(\hat{\pi})$ have been computed exactly when pricing as noted in Observation 1-$(iii.d)$. On the segment between the in-point and the out-point, one defines a *sep-point* at distance α from the out-point:

$$(\pi^{\text{sep}}, \eta^{\text{sep}}) := \alpha\,(\pi^{\text{in}}, \eta^{\text{in}}) + (1-\alpha)\,(\pi^{\text{out}}, \eta^{\text{out}})\ . \tag{29}$$

The in-out separation strategy consists in attempting to cut such sep-point. If an exact separation/pricing oracle fails to yield a separation hyperplan that cuts this sep-point, the point proves to be a valid in-point. Else, the out-point is updated. For standard in-out separation, where either the in-point or the out-point is replaced by the sep-point at each iteration, [2] proves that the distance between them tends to zero during a mis-pricing sequence.



Case A Case B Case C

Fig. 2: The three cases that can arise when attempting to cut point $(\pi^{\text{sep}}, \eta^{\text{sep}})$. Case A: $(\pi^{\text{sep}}, \eta^{\text{sep}})$ is cut-off by z^t. Case B: $(\pi^{\text{sep}}, \eta^{\text{sep}})$ is not cut but $(\pi^{\text{out}}, \eta^{\text{out}})$ is. Case C: neither $(\pi^{\text{sep}}, \eta^{\text{sep}})$, nor $(\pi^{\text{out}}, \eta^{\text{out}})$ are cut, which results in a *mis-pricing*.

The following proposition formalizes the properties common to Neame's and Wentges' smoothing schemes for column generation. Observe that the smoothing schemes described by rule (24) and (25) differ from the above standard in-out separation by the way in which the component η of the current solution is updated. Indeed, solving the separation/pricing problem yields a supporting hyperplan and a valid Lagrangian bound which is exploited in point (ii) below.

Proposition 1 *Common properties to both Neame's and Wentges' Smoothing Schemes.*
(i) *If the separation point* (π^{sep}, η^{sep}) *is cut by the inequality defined by* $z_{\pi^{sep}}$*, i.e., if* $L(\pi^{sep}) = \pi^{sep} a + (c - \pi^{sep} A) z_{\pi^{sep}} < \eta^{sep}$*, then* (π^{out}, η^{out}) *is cut off and* $z_{\pi^{sep}}$ *defines a negative reduced cost column for [*M^t*], i.e.,* $(c - \pi^{out} A) z_{\pi^{sep}} + \pi^{out} a < \eta^{out}$*, as illustrated in Case A of Figure 2.*
(ii) *In the case* (π^{sep}, η^{sep}) *is not cut, i.e., if* $L(\pi^{sep}) \geq \eta^{sep}$*, then* $(\pi^{sep}, L(\pi^{sep}))$ *defines a new in-point that may be used for the next iteration. Moreover, as* $\eta^{sep} = \alpha\,\eta^{in} + (1-\alpha)\,\eta^{out}$*, the new dual bound,* $L(\pi^{sep})$*, obtained when solving the pricing problem, improves the optimality gap at the smoothed price* $(c\,x^t - L(\tilde{\pi}^t))$ *by a factor* α*:*

$$(c\,x^t - L(\tilde{\pi}^t)) = (\eta^{out} - L(\pi^{sep})) \leq (\eta^{out} - \eta^{sep}) = \alpha\,(\eta^{out} - \eta^{in}) \leq \alpha\,(c\,x^t - L(\tilde{\pi}^{t-1}))\ , \tag{30}$$

as illustrated in Figure 3.
(iii) *The cut defined by* $z_{\pi^{sep}}$ *can cut-off* (π^{out}, η^{out}) *even if it did not cut* (π^{sep}, η^{sep})*, as illustrated in Case B of Figure 2. If it does, both the in-point and the out-point can be updated. Otherwise, failing to cut the out-point, as illustrated in Case C of Figure 2, leads to a* mis-pricing*. Then,* $(\pi^{out}, \eta^{out}) = (\pi^t, \eta^t)$ *remains solution for [*DM^{t+1}*]*

defined from [DM^t] by adding generator $z_{\pi^{sep}}$; but, under both rules (24) and (25), the smoothed prices of the next iterate get closer to kelley's prices:

$$||\tilde{\pi}^{t+1} - \pi^{t+1}|| = \alpha \, ||\tilde{\pi}^{t} - \pi^{t}|| < ||\tilde{\pi}^{t} - \pi^{t}|| \, . \tag{31}$$

Proof: Proposition 1-(i) results from the convexity of polyhedron (10-12). Proposition 1-(ii) states that if the separation point cannot be cut, it is proven feasible for polyhedron (10-12); the first inequality in bound improvement Property (30) is a simple re-writing of the assumed condition $L(\pi^{\text{sep}}) \geq \eta^{\text{sep}}$, while the second inequality is satisfied at equality for rule (24) and derives from $\eta^{\text{in}} = \hat{L} \geq L(\tilde{\pi}^{t-1})$ for rule (25). Proposition 1-(iii) observes that in the case (ii), there can be a mis-pricing or not. If there is a mis-pricing at iteration t, $\pi^{t+1} = \pi^t$ and the next sep-point can be shown to be closer to the out-point. Indeed, as $\pi^{\text{sep}} = \alpha\,\pi^{\text{in}} + (1-\alpha)\,\pi^{\text{out}}$, $||\pi^{\text{sep}} - \pi^{\text{out}}|| = \alpha \, ||\pi^{\text{in}} - \pi^{\text{out}}||$. And, at iteration $t+1$, $\tilde{\pi}^{t+1} = \pi^{\text{sep}}$, $\pi^{\text{in}} = \tilde{\pi}^t$, while $\pi^{\text{out}} = \pi^{t+1} = \pi^t$. ∎

Other smoothing schemes that differ by the rules for updating the in-point are possible. Property (30) remains valid provided $\eta^{\text{in}} \geq L(\tilde{\pi}^{t-1})$.

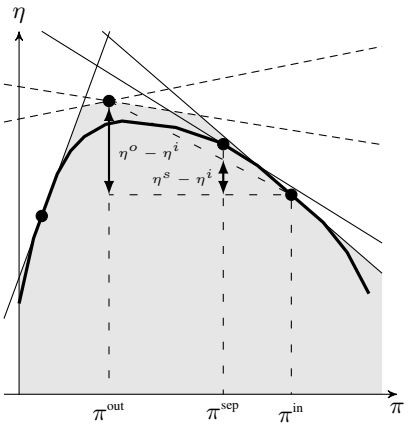


Fig. 3: In the case where $(\pi^{\text{sep}}, \eta^{\text{sep}})$ cannot be cut, then $(L(\pi^{\text{sep}}) - \eta^{\text{in}}) \geq (\eta^{\text{sep}} - \eta^{\text{in}}) = (1-\alpha)\,(\eta^{\text{out}} - \eta^{\text{in}})$, as outlined in Proposition 1-(ii).

Hence, the smoothing rules of Neame and Wentges can be understood as a projection in the π-space of the in-out separation procedure of [2], where the in-point is updated even when the sep-point is cut. The update of the η value to a valid dual bound guarantees the feasibility of the udpated in-point. In Wentges'smoothing scheme [12], the in-point is redefined as the dual incumbent at each iterate. Note however that when the separation point cannot be cut, $L(\pi^{\text{sep}}) > \hat{L}$ according to Proposition 1-(ii) and $\hat{\pi}$ is updated to π^{sep}. Thus, Wentges'smoothing conforms to the standard in-out paradigm. However, Neame smoothing scheme [8] differs from the standard in-out procedure by the fact that π^{in} is updated to π^{sep} whether or not the sep-point was cut. It can be seen as a valid variant of the in-out procedure as, even if $(\pi^{\text{sep}}, \eta^{\text{sep}})$ is not an in-point, $(\pi^{\text{sep}}, L(\pi^{\text{sep}}))$ defines an in-point that can used in the next iteration, as done implicitly in rule (24). In any case, Proposition 1 holds true for Neame smoothing scheme, as well as for Wentges. We emphasize that Proposition 1-(ii) is valid even if there is no mis-pricing. It has no equivalent for the general in-out procedure of [2] where no special component is associated with the objective value. To the best of our knowledge, such results had not been proven for Neame's smoothing scheme [8]. For

Wentges smoothing, Property (iii) was already mentioned in [12], while Property (ii), which then takes the form $(c\,x^t - L(\hat{\pi}^t)) \leq \alpha\,(c\,x^t - L(\hat{\pi}^{t-1}))$, was proven in [9], but in a more intricate manner relying on the concavity of function $L^t()$ defined in (21) and under a mis-pricing assumption.

4. α-Schedule and Convergence

Instead of using the same α for all iterations, one can define iteration-dependent values α_t. We refer to α-schedule as the procedure used to select values of α_t dynamically. Intuitively, a large α can yield deeper cut if no mis-pricing occurs, while a small α can yield large dual bound improvement if a mis-pricing occurs. But a large α resulting in a mis-pricing or a small α with no mis-pricing result in an iterate with little progress being made. The primary concern should be the overall convergence of the method, which can be guaranteed by Proposition 1. If no smoothing is used, i.e., $\alpha_t = 0\ \forall t$, the procedure is a standard Simplex based column generation for which finite convergence is proven, provided a cycle breaking rule that guarantees that each basis is visited at most once. When smoothing is used on the other hand, the same basis can remain optimal for several iterations in a sequence of mis-pricings. However, Proposition 1-(ii) provides a global convergence measure: the optimality gap $||c\,x^t - L(\tilde{\pi}^t)||$ decreases by a factor α in the case of a mis-pricing, hence the total number of mis-pricing iterations is bounded. Alternatively, Proposition 1-(iii) provides a convergence measure local to a mis-pricing sequence: $||\pi^{\text{sep}} - \pi^{\text{out}}||$ decreases during such sequence, thereby bounding the number of mis-pricings for a given LP solution π^{out}. Thus, in the line of the asymptotic convergence proof for in-out separation of [2], one can show that:

Proposition 2 *Finite convergence.*
Applying a Simplex based column generation procedure to (13-15) while pricing on smoothed prices as set in (26), using either Neame (24) *or Wentges* (25)*'s rule, converges to an optimal solution after a finite number of iterations, i.e., for some $t \in I\!N$, $(\pi^t, \eta^t) = (\pi^*, \eta^*)$, where (π^*, η^*) is an optimal solution to (10-12).*

Proof: The number of columns that can be generated is finite. A column, once added, cannot price out negatively: the associated dual constraint is satisfied by both the in-point and out-point (hence by the sep-point). However, in the case of a mis-pricing, there can be several iterations where no new column arises as a solution to the pricing problem. So the proof relies on bounding the number of such iterations. Consider a sequence of mis-pricing iterations, starting with $\pi^{\text{in}} = \tilde{\pi}^0$. Then, at iteration $t+1$,

$$||\tilde{\pi}^{t+1} - \pi^{\text{out}}|| = \alpha_t\,||\tilde{\pi}^t - \pi^{\text{out}}|| = \ldots = \mathit{\Pi}_{\tau=0}^{t}\alpha_\tau\,||\tilde{\pi}^0 - \pi^{\text{out}}|| \text{ and}$$

$$||c\,x^{t+1} - L(\tilde{\pi}^{t+1})|| \leq \alpha_t\,||c\,x^t - L(\tilde{\pi}^t)|| \leq \ldots \leq \mathit{\Pi}_{\tau=0}^{t}\alpha_\tau\,||c\,x^0 - L(\tilde{\pi}^0)||\,.$$

Hence, since $\alpha_\tau \in [0,1)$, $\mathit{\Pi}_{\tau=0}^{t}\alpha_\tau \to 0$, $\tilde{\pi}^t \to \pi^{\text{out}}$, and $L(\tilde{\pi}^t) \to c\,x^t = \eta^t = \eta^{\text{out}}$, as $t \to \infty$. For any given precison level $\epsilon > 0$, $\exists t$ such that $||\tilde{\pi}^t - \pi^{\text{out}}|| \leq \epsilon$ and $||\eta^t - L(\tilde{\pi}^t)|| \leq \epsilon$ and the master is considered as optimized. Indeed, as $L(\tilde{\pi}^t) \leq \eta^* \leq \eta^t$, $||\eta^t - \eta^*|| \leq ||\eta^t - L(\tilde{\pi}^t)||$. Moreover, since there is a finite number of possible values for η^t, $\exists \epsilon > 0\ : ||\eta^t - L(\tilde{\pi}^t)|| \leq \epsilon \Rightarrow \eta^t = \eta^*$. The associated price vector, π^t, is optimal, i.e. $\pi^t = \pi^*$. ∎

Asymptotically convergent algorithms might not be suitable for practical purposes. For instance, consider setting $\alpha = 0.8$ for all t, as illustrated in the left of Figure 4. Then, the distance reduction in a mis-pricing sequence becomes small very quickly. In practice, it would be better to choose an α-schedule such that $\tilde{\pi}^t = \pi^t$ after

a small number of mis-pricing iterations t, as illustrated in the right of Figure 4. Given a static baseline α, we propose as outlined on the left side in Table 1, to adapt α_t during a mis-pricing sequence in such a way that $(1 - \Pi_{\tau=0}^{k} \alpha_\tau) = k * (1 - \alpha)$. Hence, $\alpha_t = 0$ after $k = \left\lceil \frac{1}{(1-\alpha)} \right\rceil$ mis-pricing iterations, at which point smoothing stops, as $\tilde{\pi}^t = \pi^t$, which forces the end of a mis-pricing sequence.

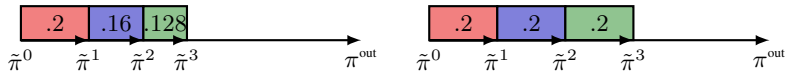


Fig. 4: At the t-th mis-pricing iteration, $||\tilde{\pi}^0 - \pi^{\text{out}}||$ is cut by a factor $(1 - \Pi_{\tau=0}^{t} \alpha_\tau)$.

So far we assumed a static baseline α provided as an input. Let us now consider how the user could be free from having to tune α for his application. In deriving an auto-adaptive α-schedule, one could consider using high α while the out-point is believed to be a bad approximation, and reducing α as the method converges, which is measured by smaller gaps $|\eta^t - \hat{L}|$, and the purpose becomes to prove optimality. Alternatively, one could rely on local information, as we do. We propose to decrease α when the sub-gradient at the sep-point indicates that a larger step from the in-point would further increase the dual bound (i.e., when the angle of the ascent direction, g^{sep}, as defined in (22), and the direction $(\pi^{\text{out}} - \pi^{\text{in}})$ is less than $90°$), and vice versa. We outline this procedure on the right side in Table 1, while Figure 5 offers an illustration. Functions for increasing and decreasing α are illustrated in Figure 6: $f_{\text{incr}}(\alpha_t) = \alpha_t + (1 - \alpha_t) \cdot 0.1$, while $f_{\text{decr}}(\alpha_t) = \alpha_t / 1.1$ if $\alpha_t \in [0.5, 1)$, and $f_{\text{decr}}(\alpha_t) = \max\{0, \alpha_t - (1 - \alpha_t) \cdot 0.1\}$, otherwise.

Table 1: α-schedule in a mis-pricing sequence for a given initial α (on the left) and dynamic α-schedule based on sub-gradient information for a given intial α (on the right).

Step 0: $k \leftarrow 1$, $\pi^0 \leftarrow \pi^{\text{in}}$	Step 0: Let $\alpha_0 \leftarrow \alpha$, $t \leftarrow 0$.
Step 1: $\tilde{\alpha} \leftarrow [1 - k * (1 - \alpha)]^+$	Step 1: Call pricing on $\pi^{\text{sep}} = \alpha_t \pi^{\text{in}} + (1 - \alpha_t)\pi^{\text{out}}$.
Step 2: $\pi^{\text{sep}} = \tilde{\alpha}\, \pi^0 + (1 - \tilde{\alpha})\, \pi^{\text{out}}$	Step 2: If a mispricing occurs, start the mispricing schedule.
Step 3: $k \leftarrow k + 1$	Step 3: Else, let g^{sep} be the sub-gradient in sol $z_{\pi^{\text{sep}}}$.
Step 4: call the pricing oracle on π^{sep}	Step 4: If $g^{\text{sep}}(\pi^{\text{out}} - \pi^{\text{in}}) > 0$, $\alpha_{t+1} \leftarrow f_{\text{incr}}(\alpha_t)$; otherwise, $\alpha_{t+1} \leftarrow f_{\text{decr}}(\alpha_t)$.
Step 5: if a mis-pricing occurs, goto Step 1; else, let $t \leftarrow t + 1$, solve the master and goto Step 0.	Step 5: Let $t \leftarrow t + 1$, solve the master and goto Step 1.

5. Hybridization with an ascent method

With a pure smoothing technique, the price vector is defined by taking a step $(1 - \alpha)$ from the in-point in the direction of the out-point: $\pi^{\text{sep}} = \pi^{\text{in}} + (1 - \alpha)(\pi^{\text{out}} - \pi^{\text{in}})$. Here, we consider modifying the direction $(\pi^{\text{out}} - \pi^{\text{in}})$ by twisting it towards the direction of ascent observed in π^{in}. The resulting method can be viewed as a hybridization of column generation with a sub-gradient method. When Wentges's rule (25) is used, the resulting hybrid method

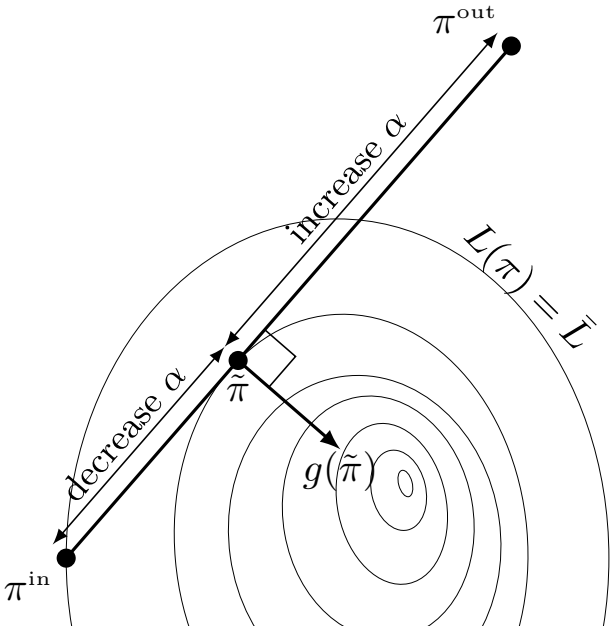


Fig. 5: Auto-adaptive α-schedule: condition for increasing or decreasing α depending on whether the π^{sep} is below or beyond the threshold value $\tilde{\pi}$.

is related to the Volume algorithm [1] where π^t is obtained by taking a step from $\hat{\pi}$ in a direction that combines previous iterate information with the current sub-gradient. However, contrary to the Volume algorithm, our purpose here is not to derive the next π^t iterate, but simply to bring a correction to the price vector that is used in the pricing procedure.

Table 2: Directional smoothing with parameter β.

Step 1: $\tilde{\pi} = \pi^{\text{in}} + (1-\alpha)(\pi^{\text{out}} - \pi^{\text{in}})$
Step 2: $\pi^g = \pi^{\text{in}} + \frac{g^{\text{in}}}{\|g^{\text{in}}\|} \|\pi^{\text{out}} - \pi^{\text{in}}\|$
Step 3: $\rho = \beta\pi^g + (1-\beta)\pi^{\text{out}}$
Step 4: $\pi^{\text{sep}} = \left(\pi^{\text{in}} + \frac{\|\tilde{\pi} - \pi^{\text{in}}\|}{\|\rho - \pi^{\text{in}}\|} (\rho - \pi^{\text{in}})\right)^+$

The hybrid procedure, that we call *directional smoothing*, is outlined in Table 2 and illustrated in Figure 7. Let g^{in} denote the sub-gradient associated to oracle solution $z_{\pi^{\text{in}}}$. In Step 1, $\tilde{\pi}$ is computed by applying smoothing. In Step 2, π^g is computed as the point located on the steepest ascent direction at a distance from π^{in} equal to the distance to π^{out}. In Step 3, a rotation is performed, defining target ρ as a convex combination between π^g and π^{out}. Then, in Step 4, the sep-point is selected in direction $(\rho - \pi^{\text{in}})$ at the distance from π^{in} equal to $\|\tilde{\pi} - \pi^{\text{in}}\|$ and it is projected on the positive orthant. As is the case with non-directional smoothing, using modified dual prices can result in mis-pricing. When this arises, we switch off directional smoothing by setting $\beta = 0$ in the next iteration. Apart for mis-pricing, directional smoothing can be implemented with a fixed value of parameter β. However,

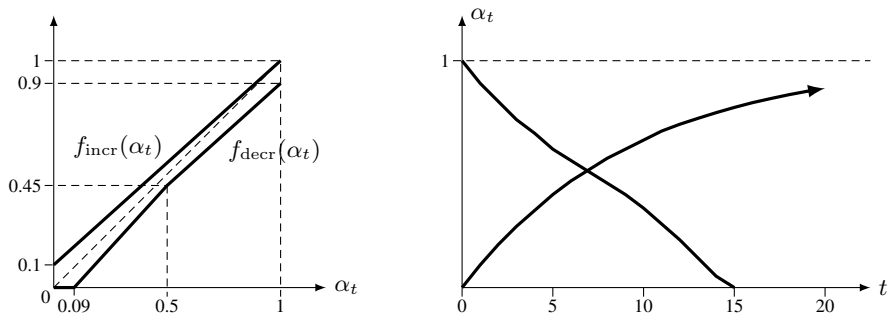


Fig. 6: Functions for increasing or decreasing α are presented on the left side. It is essential to have a non negligible decreasing factor when α is close to 1 to avoid slow adjustments. The right side shows how the value of α changes when α increases of decreases on each iteration using these functions.

computational experiments showed that the larger the angle γ between vectors $(\pi^{\text{out}} - \pi^{\text{in}})$ and $(\pi^g - \pi^{\text{in}})$, the smaller the value for β should be. Indeed, if the angle γ is large, then twisting the direction is likely to lead to a mis-pricing. Our proposal is to use an adaptive β-schedule by setting $\beta = \cos\gamma$. As γ is always less than $90°$, since vector $(\pi^{\text{out}} - \pi^{\text{in}})$ is an ascent direction, $\beta \in [0, 1]$.

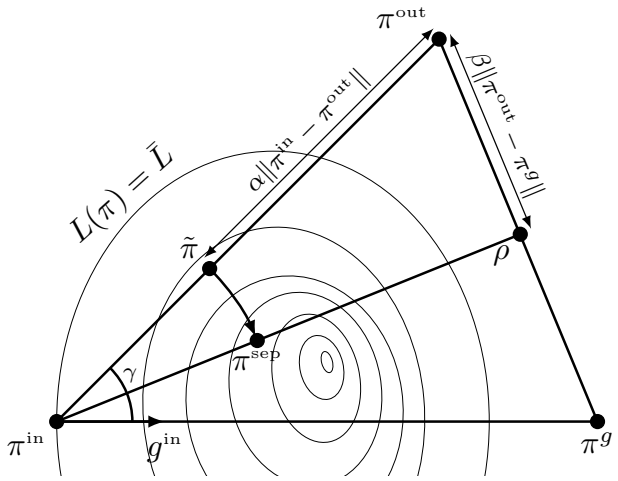


Fig. 7: Directional Smoothing: combining sub-gradient and Wentges smoothing

6. Numerical tests

In the experiments we describe next, we assess numerically the stabilization effect of applying Wentges smoothing with static α-schedule versus auto-adaptive schedule starting with $\alpha_0 = 0.5$. Additionally, we estimate the effect of using directional smoothing, with static and auto-adaptive value of parameter β, in combination with Wentges smoothing. The experiments are conducted on the following problems and instance sets (further described in the appendix):

- **Parallel Machine Scheduling:** 30 instances generated in the same way as in the OR-Library with number of machines in $\{1, 2, 4\}$ and jobs in $\{50, 100, 200\}$.
- **Generalized Assignment:** 18 OR-Library instances of types D and E with number of agents in $\{5, 10, 20, 40\}$ and jobs in $\{100, 200, 400\}$.
- **Multi-Echelon Small-Bucket Lot-Sizing:** 17 randomly generated instances varying by the number of echelons in $\{1, 2, 3, 5\}$, items in $\{10, 20, 40\}$, and periods in $\{50, 100, 200, 400\}$.
- **Bin Packing:** 12 randomly generated instances with number of items in $\{400, 800\}$ and average number of items per bin in $\{2, 3, 4\}$.
- **Capacitated Vehicle Routing:** 21 widely used instances from the literature of types A, B, E, F, M, P with 50-200 clients and 4-16 vehicles.

For each instance, we determine experimentally the best static α-value for which the master LP solution time by column generation with Wentges smoothing is minimum, by testing all α values in $\{0.05, 0.1, \ldots, 0.95\}$, as illustrated in Figures 8 and 9. The first columns of Table 3 report respectively the geometric mean of CPU time without smoothing on a Dell PowerEdge 1950 (32Go, Intel Xeon X5460, 3.16GHZ) and the range of best α-values that vary a lot from one instance to the next and between applications. In the other columns of Table 3, we compare tuned and self-adjusting smoothing to standard column generation without any smoothing. Next, in Table 4, we compare performance with and without the extra directional feature, using both a static parameter β (the best in the set $\{0.05, 0.1, 0.2, 0.3\}$) and an adaptive β. Thus, in total we compare 6 variants of column generation: (i) without any stabilization ($\alpha = 0$, $\beta = 0$), (ii) with static Wentges stabilization ($\alpha = best$, $\beta = 0$), (iii) with auto-adaptive Wentges stabilization ($\alpha = auto$, $\beta = 0$), (iv) with combined static Wentges and directional stabilization ($\alpha = best$, $\beta = best$), (v) with combined static Wentges and adaptive directional stabilization ($\alpha = best$, $\beta = auto$), and (vi) with combined adaptive Wentges and directional stabilization ($\alpha = auto$, $\beta = auto$). In the tables, we report ratios of geometric means for the following statistics: `It` is the number of iterations in column generation; `Cl` is the number of columns generated; `T` is the solution time. The last columns of Table 4 summarizes the overall performance of smoothing. Note that smoothing improves solution times by a larger factor than the number of iterations, in spite of the potentially harder pricing subproblems. Figure 10 shows how α and β values evolve in the course of the algorithm using the self-adjusting schemes.

	$\alpha = 0$	Best α	Ratio $\frac{\alpha = 0}{\alpha = best}$			Ratio $\frac{\alpha = 0}{\alpha = auto}$			Ratio $\frac{\alpha = best}{\alpha = auto}$		
Problem	time	`Range`	`It`	`Cl`	`T`	`It`	`Cl`	`T`	`It`	`Cl`	`T`
Generalized Assignment	98	[0.5,0.95]	3.37	3.50	4.46	3.36	3.75	4.57	1.00	1.07	1.03
Lot-Sizing	88	[0.4,0.95]	2.26	2.93	3.31	2.51	3.66	4.58	1.11	1.25	1.38
Machine Scheduling	33	[0.65,0.9]	2.30	2.30	3.04	2.29	2.29	2.98	1.00	1.00	0.98
Bin Packing	7.9	[0.75,0.95]	1.54	1.48	1.79	1.49	1.45	1.65	0.97	0.98	0.92
Vehicle Routing	6.3	[0.2,0.8]	1.32	1.40	1.37	1.15	1.46	1.28	0.88	1.04	0.94

Table 3: Stabilization effect of Wentges smoothing with static versus auto-adaptive α: showing geometric means.

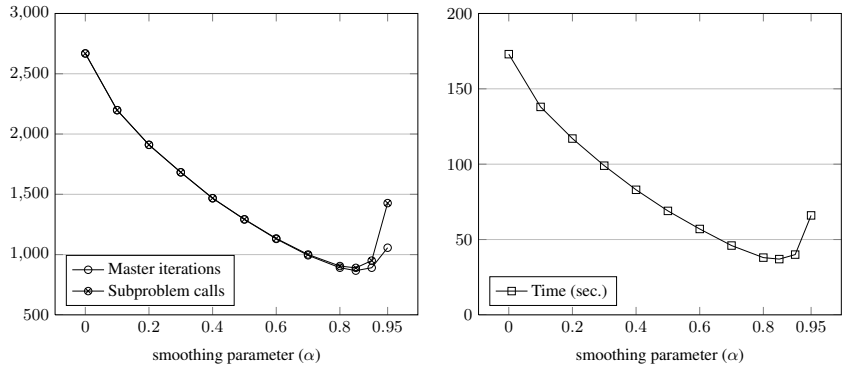


Fig. 8: Sensitivity to a static α for machine scheduling random instances with 50 jobs per machine. Although the figures seem to indicate that a 0.85 α-value is best, such tuning depends highly on the application and differs a lot from one instance to the next as reflected by the range of best α-values reported in Table 3.

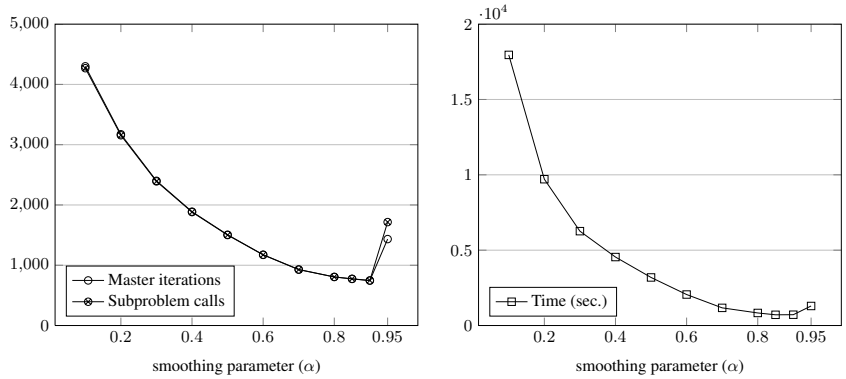


Fig. 9: Sensitivity to a static α for generalized assignment instances from the OR-Library C, D, E class with 40 jobs per machine.

Conclusion

In this paper, we have specified the link between column generation stabilization by smoothing and in-out separation. We also extended the in-out convergence proof for Neame's and Wentges' smoothing schemes, deriving an extra global convergence property on the optimality gap. These results trivially extend to the case of multiple subproblems. On the practical side, our numerical results confirm the effectiveness of smoothing and show that it can be implemented in a way that does not require parameter tuning. Our hard coded initialization and dynamic auto-adaptive scheme based on local sub-gradient information experimentally matches or improves the performance of the best user tuning as revealed in Table 3. The extra directional twisting feature is shown in Table 4 to bring further performance improvement in the non-identical subproblem case. When there are identical subproblems, as in Bin Packing and Vehicle Routing, sub-gradient information is aggregated and hence probably less pertinent. The adaptive setting of parameter β outperforms the static value strategy leading to a generic parameter-tuning-free im-

	Ratio $\frac{\alpha = best, \beta = 0}{\alpha, \beta = best}$			Ratio $\frac{\alpha = best, \beta = 0}{\alpha = best, \beta = auto}$			Ratio $\frac{\alpha = best, \beta = 0}{\alpha, \beta = auto}$			Ratio $\frac{\alpha, \beta = 0}{\alpha, \beta = auto}$	
Problem	`It`	`Cl`	`T`	`It`	`Cl`	`T`	`It`	`Cl`	`T`	`It`	`T`
Generalized Assignment	1.11	1.08	1.93	1.35	1.38	1.95	1.48	1.63	2.25	5.00	10.03
Lot-Sizing	1.17	1.27	1.50	1.32	1.39	1.61	1.37	1.69	1.83	3.09	6.06
Machine Scheduling	0.94	0.86	0.91	1.04	1.03	1.12	1.10	1.10	1.21	2.53	3.68
Bin Packing	0.95	0.95	0.94	1.03	1.02	0.98	1.04	1.03	0.96	1.60	1.72
Vehicle Routing	0.90	0.95	0.92	0.94	0.99	0.97	0.83	1.03	0.92	1.09	1.25

Table 4: Extra stabilization effect when applying directional smoothing

plementation. This work can hopefully inspire a renewed interest in cut separation strategies, possibly developing in-out separation with a self-adjusting parameter scheme and a gradient direction twist.

Aknowledments: This research was supported by the associated team program of INRIA through the SAMBA project. The remarks of J-P. Richard and anonymous referees were of great value to improve our draft.

References

1. F. Barahona and R. Anbil. The Volume Algorithm: Producing Primal Solutions with a Subgradient Method. *Math. Program. A*, 87(1):385–399, 2000.
2. W. Ben-Ameur and J. Neto. Acceleration of Cutting-Plane and Column Generation Algorithms: Applications to Network Design. *Networks*, 49(1):3–17, 2007.
3. O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of Bundle and Classical Column Generation. *Math. Program. A*, 113(2):299–344, 2008.
4. O. du Merle, D. Villeneuve, J. Desrosiers and P. Hansen. Stabilized Column Generation. *Discrete Math.*, 194:229-237, 1999.
5. M. Fischetti and D. Salvagnin. An In-Out Approach to Disjunctive Optimization. *Lect. N. in Comp. Sc.*, 6140:136–140, 2010.
6. J-L. Goffin and J-Ph. Vial. Convex Nondifferentiable Optimization: A Survey Focused on the Analytic Center Cutting Plane Method. *Optimization Methods and Software*, 17(5):805–867, 2002.
7. C. Lee and S. Park. Chebyshev Center Based Column Generation. *Discrete Applied Mathematics*, 159(18), 2011.
8. P.J. Neame. *Nonsmooth Dual Methods in Integer Programming*. PhD thesis, 1999.
9. A. Pessoa, E. Uchoa, M. Poggi de Aragão, and R. Rodrigues. Exact Algorithm over an Arc-Time-Indexed Formulation for Parallel Machine Scheduling Problems. *Math. Prog. Computation*, 2:259–290, 2010.
10. F. Vanderbeck and M.W.P. Savelsbergh. A Generic View of Dantzig-Wolfe Decomposition in Mixed Integer Programming. *Operations Research Letters*, 34(3):296–306, 2006.
11. F. Vanderbeck. Implementing Mixed Integer Column Generation. In *Column Generation*, Kluwer (2005).
12. P. Wentges. Weighted Dantzig-Wolfe Decomposition for Linear Mixed-Integer Programming. *Int. Trans. O. R.*, 4:151-162, 1997.

Appendix: Details on the numerical test setting

For all problems, Cplex 12.4 was used as an LP solver to solve the restricted master on every iteration. No time limit was set. For all instances, the linear programming relaxation of the master was solved to optimality.

Machine Scheduling

The instances of the parallel machine scheduling problem were generated similarly to single machine weighted tardiness scheduling instances in OR-Library. The objective function is the total weighted tardiness (this problem is denoted as $P \mid\mid \sum w_j T_j$). Instance size is determined by a triple $(n, m, p_{\max})$, where n is the number of jobs, m is the number of machines, and $p_{\max}$ is the maximum processing time of jobs. Instances are generated using the procedure of Potts and van Wassenhove (1985): integer processing times p_j are uniformly distributed in interval $[1, p_{\max}]$ and integer weights w_j in $[1, 10]$ for jobs j, $j = 1, \ldots, n$, while integer due dates have been generated from a uniform distribution in $[P(1 - TF - RDD/2)/m, P(1 - TF + RDD/2)/m]$, where $P = \sum_j p_j$, TF is a tardiness factor, and RDD is a relative range of due dates. We generated instances of sizes determined by triples $(1, 50, 100)$, $(1, 100, 100)$, $(2, 50, 100)$, $(2, 100, 100)$, $(4, 100, 100)$, $(4, 200, 100)$. For each size, one instance was generated for each of the following pairs (TF, RDD): $(0.2, 0.2)$, $(0.4, 0.4)$, $(0.6, 0.6)$, $(0.8, 0.8)$, $(1.0, 1.0)$. Therefore, in total, 30 instances were generated. For solving the pricing subproblem, a standard dynamic programming algorithm of complexity $O(nP)$ has been used.

Generalized Assignment

The instances of the generalized assignment problem were taken from OR-Library and from website of Yagiura (`http://www.al.cm.is.nagoya-u.ac.jp/~yagiura/gap/`). Each instance name is in the $C-m-n$ format, where C is instances class, m is the number of machines (or agents), and n is the number of jobs (or tasks). We used the following instances $D-10-100$, $E-10-100$, $D-20-200$, $E-20-200$, $D-40-400$, $E-40-400$, $D-5-100$, $E-5-100$, $D-10-200$, $E-10-200$, $D-20-400$, $E-20-400$, $D-5-200$, $E-5-200$, $D-10-400$, $E-10-400$. For solving the binary knapsack pricing subproblem, we used the *minknap* solver of Pisinger: `http://www.diku.dk/~pisinger/codes.html`.

Multi-Echelon Lot-Sizing

The instances of the multi-echelon lot-sizing problem have been generated randomly using uniform distribution. The size of an instances is determined by triple (E, K, T), where E is the number of echelons, K is the number of items, and T is the number of periods. Setup costs are in interval $[20, 100]$, production costs are zero, and storage costs h_e^k are generated as $h_{e-1}^k + \gamma$, where γ is in interval $[1, 5]$, and $h_0^k = 0$. For each period, there is a positive demand for 3 items on average. Demands are in interval $[10, 20]$. We generated one instance for each of the following size triples: $(1, 20, 100)$, $(1, 40, 200)$, $(1, 10, 100)$, $(1, 20, 200)$, $(1, 40, 400)$, $(2, 20, 50)$, $(2, 10, 50)$, $(2, 20, 100)$, $(2, 10, 100)$, $(3, 20, 50)$, $(3, 10, 50)$, $(3, 20, 100)$, $(3, 10, 100)$, $(5, 20, 50)$, $(5, 10, 50)$, $(5, 20, 100)$, $(5, 10, 100)$. To solve the pricing problem for the one echelon instances, a standard dynamic programming algorithm of complexity $O(T^2)$ is used. For instances with multiple echelons, a standard dynamic programming algorithm of complexity $O(ET^4)$ is used.

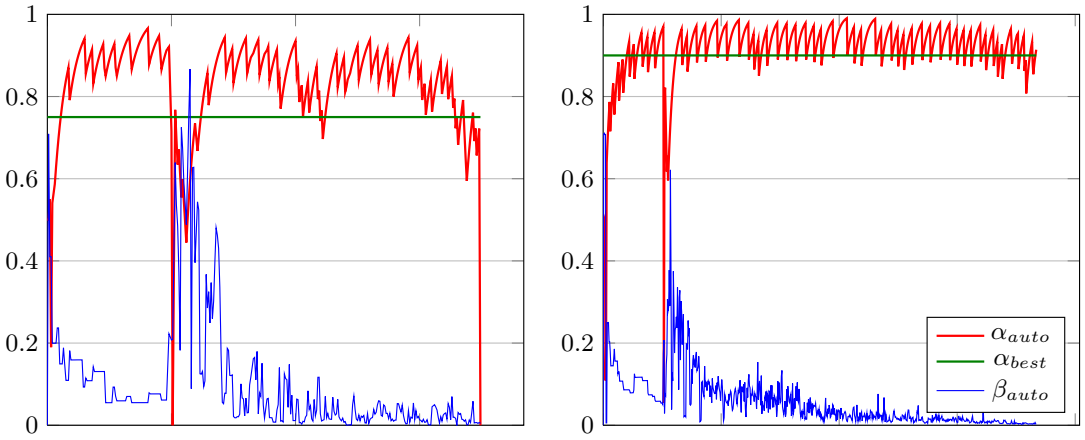
Bin Packing

The instances of the bin packing problem have been generated randomly using uniform distributions. Instance classes "a2", "a3", and "a4" (the number refers to the average number of items per bin) contain instances with bin capacity equal to 4000 where item sizes are in intervals $[1000, 3000]$, $[1000, 1500]$, and $[800, 1300]$, respectively. We generated four instances for each class (two instances with 400 items and two instances with 800 items).

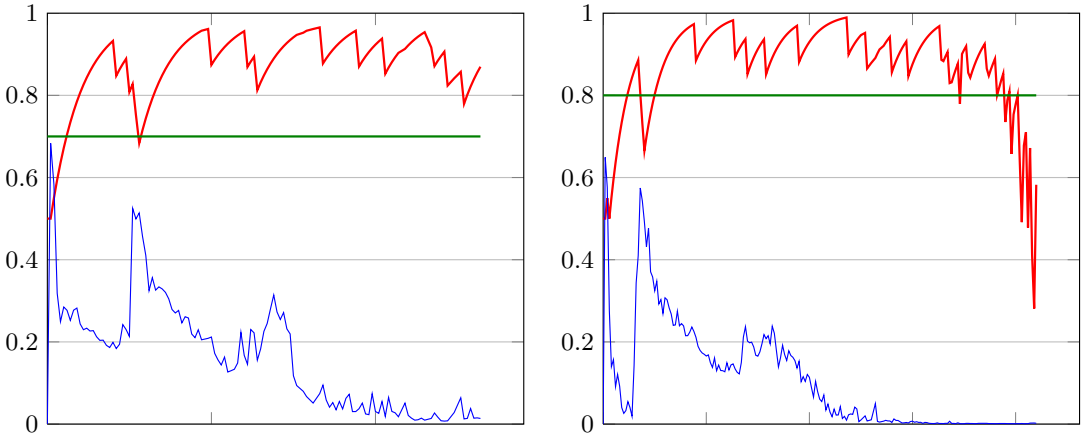
For solving the binary knapsack pricing subproblem, we used the *minknap* solver of Pisinger as for Generalized Assignment.

.1 Capacitated Vehicle Routing

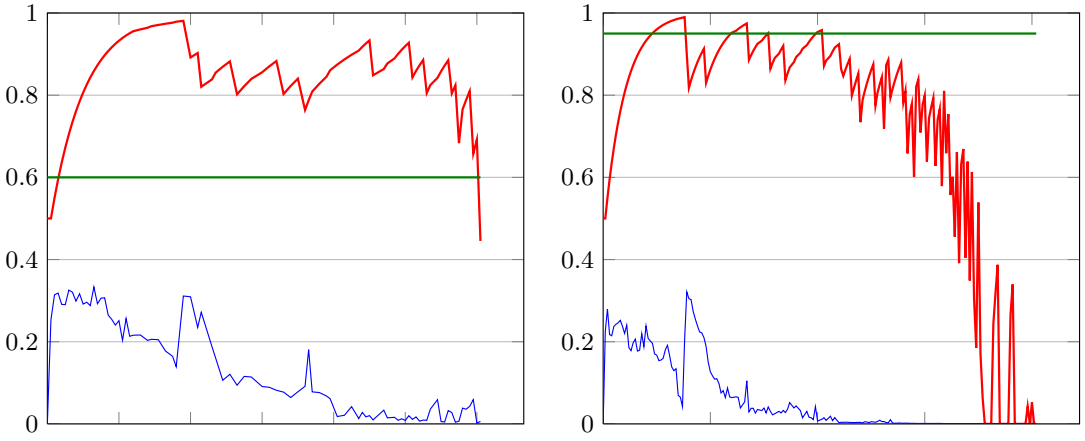
The instances of the capacitated vehicle routing problem are classic symmetric instances. They can be downloaded, for example, at `neo.lcc.uma.es/vrp/vrp-instances/`. Names of the instances are in the format $C-nN-kK$, where C is the instance class, N is the number of customers, K is the number of identical vehicles. We used instances $A-n63-k10$, $A-n64-k9$, $A-n69-k9$, $A-n80-k10$, $B-n50-k8$, $B-n68-k9$, $E-n51-k5$, $E-n76-k7$, $E-n76-k8$, $E-n76-k10$, $E-n76-k14$, $E-n101-k8$, $F-n72-k4$, $M-n121-k7$, $M-n151-k12$, $M-n200-k16$, $P-n50-k8$, $P-n70-k10$, $P-n76-k5$, $P-n101-k4$. For solving the pricing route generation problem, an algorithm of Baldacci et al. ("New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem", Oper. Res. 59(5), 2011.) is used, which generates ng-routes. For each customer vehicle, we select the 3 closest customers.



Machine Scheduling m4-j100-k1 Machine Scheduling m4-j100-k19



Generalized Assignment E-20-200 Generalized Assignment E-10-200



Lot Sizing 1-40-200 Lot Sizing 2-10-100

Fig. 10: Representation of α and β values in the course of the algorithm using the self-adjusting schemes, versus the best static α value, for representative instances. One can identify two stages in the solution process. The first stage ends with the first major pick, when the (possibly artificial) columns used to initialize the restricted master get out of the solution. A steep decreasing of the α value represents a mis-pricing sequence, while a slower decrease or increase represents an adjustment based on subgradient information. The β value gets small when the gradient direction in $\hat{\pi}$ makes a large angle with the direction to π^{out}. Observe the relative coherence between best static α value and that of the self-adjusting scheme in the first 5 instances. For the last Lot Sizing instance, the self-adjusting scheme is much more efficient than a static α. In this instance, a large α is desirable at the beginning while a small α value is best towards the end. This desirable setting cannot be approximated by a fixed α.