Comput Optim Appl (2009) 44: 117–138
DOI 10.1007/s10589-007-9149-2

On solving the Lagrangian dual of integer programs via an incremental approach

Manlio Gaudioso · Giovanni Giallombardo · Giovanna Miglionico

Received: 16 October 2006 / Revised: 26 October 2007 / Published online: 8 November 2007

Abstract The Lagrangian dual of an integer program can be formulated as a min-max problem where the objective function is convex, piecewise affine and, hence, nonsmooth. It is usually tackled by means of subgradient algorithms, or multiplier adjustment techniques, or even more sophisticated nonsmooth optimization methods such as bundle-type algorithms.

Recently a new approach to solving unconstrained convex finite min-max problems has been proposed, which has the nice property of working almost independently of the exact evaluation of the objective function at every iterate-point.

In the paper we adapt the method, which is of the descent type, to the solution of the Lagrangian dual. Since the Lagrangian relaxation need not be solved exactly, the approach appears suitable whenever the Lagrangian dual must be solved many times (e.g., to improve the bound at each node of a branch-and-bound tree), and effective heuristic algorithms at low computational cost are available for solving the Lagrangian relaxation.

We present an application to the Generalized Assignment Problem (GAP) and discuss the results of our numerical experimentation on a set of standard test problems.

Keywords Incremental algorithms · Convex minimization · Finite min-max · Lagrangian relaxation · Nonsmooth optimization · Bundle methods

This research has been partially supported by the Italian "Ministero dell'Università e della Ricerca Scientifica," under PRIN2005 research project *"Numerical methods for global optimization and for some classes of nondifferentiable optimization"* (2005017083_002).

M. Gaudioso · G. Giallombardo (✉) · G. Miglionico
Dipartimento di Elettronica Informatica e Sistemistica, Università della Calabria, 87036 Rende (CS), Italy
e-mail: giallo@deis.unical.it

M. Gaudioso
e-mail: gaudioso@deis.unical.it

G. Miglionico
e-mail: gmiglionico@deis.unical.it

1 Introduction

The objective of the paper is to extend a recently proposed incremental method [6] for solving convex finite min-max problems to the solution of the Lagrangian dual [14] of integer programs of the max-type.

The new incremental method presented in [6] is devoted to solve problems of the min-max type by working on the basis of "incomplete knowledge" of the objective function. In particular, at each step of the minimization process it is not necessarily required to calculate the exact value of the objective function (the maximum), but possibly only one of the functions that are involved into the maximization process. This can be very useful whenever the Lagrangian dual of an integer program is to be solved. The Lagrangian dual of an integer program of the max-type is in fact a min-max problem, where the evaluation of the objective function (the inner maximization) requires solution of the Lagrangian relaxation of the original integer program. In our approach such a problem need not be solved exactly, allowing us to use heuristic techniques instead of exact algorithms in dealing with the relaxed problem, without impairing the possibility of finding the exact solution to the Lagrangian dual, if required.

The application of the method appears suitable whenever the Lagrangian relaxation of the original integer program is not solvable in polynomial time, but there exist heuristic algorithms at low computational cost for obtaining a good feasible solution to it.

The method can be used, of course, to obtain the solution of the Lagrangian dual, but, also, can be embedded into any enumerative approach to tackle the original integer program, when at each node of the enumeration tree the Lagrangian relaxation is the tool adopted to find an upper bound, and the quality of the bound is then improved through some appropriate multiplier updating technique. In the latter case subgradient methods, or multiplier adjustment techniques, or more sophisticated nonsmooth optimization algorithms, such as bundle-type methods, are usually adopted. All such methods require exact solution of the Lagrangian relaxation for each configuration of the multipliers, which is not always the case for the proposed method.

In order to prove effectiveness of our approach, as well as to show its practical use, we will particularly concentrate our attention on the Lagrangian dual of the Generalized Assignment Problem (GAP), an extensively studied integer program known to be NP-hard. In fact, a lot of exact and heuristic algorithms have been developed in the last thirty years. A survey of algorithms developed until 1992 can be found in [3]. More recent work on these topics can be found in [12, 15–17].

Exact algorithms designed for GAP are often of the branch-and-bound type. In this context it is very important to find good bounds by using different types of relaxations. Moreover a dual descent method is necessary to tighten the bound obtained at each node, in order to possibly discard the related subproblem. One of the most used algorithm in this context is the subgradient method, but also "ad hoc" heuristic algorithms have been developed for GAP, one of the most important being the multiplier adjustment method presented in [5]. An extensive description of the Lagrangian relaxations of GAP and relative algorithms can be found in [11].

The paper is organized as follows. In Sect. 2, to make the paper selfcontained, the incremental algorithm proposed in [6], which falls in the class of bundle methods

for convex nonsmooth optimization problems [10], is shortly described. We refer the interested reader to [2] for a general discussion on the online (incremental) approach, and to [9, 13] for the analysis of the convergence properties and for an extensive bibliography. In Sect. 3 we particularize our algorithm in order to make it suitable for solving a Lagrangian dual of an integer program of the max type. Next, in Sect. 4, we show practical application of the method for solving the Lagrangian dual of GAP. Finally, in Sect. 5 we report on the numerical results obtained by applying the method to some standard GAP test problems.

For the sake of readability, we have adopted throughout the paper notations as close as possible to the ones commonly adopted in the literature related to each different context involved. Consequently, when confusion cannot occur, the same symbol has been used, in different parts of the paper, with fairly different meanings.

2 An incremental method for min-max problems

As previously announced, first we briefly review the structure of the incremental method presented in [6], along with some underlying theoretical results. The method is of the *bundle* type and is based on the cutting-plane approach to minimize convex functions. It is particularly tailored to solve the following unconstrained minimization problem of the min-max type

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the pointwise maximum over all the functions $f_i : \mathbb{R}^n \mapsto \mathbb{R}$, with $i \in I \stackrel{\triangle}{=} \{1, \ldots, m\}$, that is

$$f(x) \stackrel{\triangle}{=} \max_{i\in I} f_i(x).$$

Each function f_i is said a *component functions* of f, while the set I is referred to as the *function index set*. In the following we will assume that each f_i is convex and not necessarily differentiable.

Suppose a set of points x_k, $k = 1, \ldots, r$, is given and that at each point x_k just one of the component functions, say the one indexed by $i_k \in I$, is evaluated along with a corresponding subgradient $g_{i_k} \in \partial f_{i_k}(x_k)$. On the basis of the information provided by the set of triplets

$$\{(x_k, f_{i_k}(x_k), g_{i_k}) : k = 1, \ldots, r\},$$

the following cutting-plane model of f is generated

$$\hat{f}_r(x) = \max_{1\le k\le r} \{f_{i_k}(x_k) + g_{i_k}^\top (x - x_k)\}, \tag{2}$$

which is the pointwise maximum of the affine functions $f_{i_k}(x_k) + g_{i_k}^\top (x - x_k)$, for $k = 1, \ldots, r$. The model function $\hat{f}_r(x)$ is said a *partial cutting-plane function* since, unlike the standard cutting-plane function, it is not necessarily obtained via the linearization at each point x_k of the maximal component function, but, instead, on the basis of only one component function, no matter how it is singled out.

As a consequence of both convexity of the component functions and the definition of f, the linearization of each f_i is still a lower affine approximation of f. This in turn implies that $\hat{f}_r(x)$ is a convex piecewise-affine lower approximation of f, the main difference with the standard cutting-plane function being in the fact that the partial cutting-plane function need not interpolate the original function f at the points $x_k, k = 1, \ldots, r$. The basic idea is then to calculate the next trial point x_{r+1} as a minimizer of the partial cutting-plane function.

To make the approach work in practice, we resort to the machinery typical of bundle methods for minimizing convex functions. In particular, when the set of triplets $\{(x_k, f_{i_k}(x_k), g_{i_k}) : k = 1, \ldots, r\}$, is available, we select from among the points $x_1, \ldots, x_r$ an estimate y of the minimizer that we refer to as the current *stability center*, namely, the point from which the next displacement is to be calculated.

By letting $x = y + d$, the function $\hat{f}_r$ can be rewritten in the form rooted at y:

$$\hat{f}_r(y + d) = \max_{1 \le k \le r} \{g_{i_k}^\top d + \beta_k\},$$

where β_k is the value at point y of the linearization of f_{i_k} rooted at x_k:

$$\beta_k \stackrel{\triangle}{=} f_{i_k}(x_k) + g_{i_k}^\top (y - x_k), \quad k = 1, \ldots, r. \tag{3}$$

Now, using a notation widely adopted in the nonsmooth optimization literature, we define the "bundle" $\mathcal{B}_r$ as the following set of elements:

$$\mathcal{B}_r \stackrel{\triangle}{=} \{(x_k, f_{i_k}(x_k), g_{i_k}, \beta_k), \ \forall k \in \mathcal{I}_r\},$$

where the index set $\mathcal{I}_r \stackrel{\triangle}{=} \{1, \ldots, r\}$ is the *bundle index set*. Assuming that the bundle $\mathcal{B}_r$ is currently available, a new point x_{r+1} can be calculated by setting

$$x_{r+1} = x_r + d_r,$$

where d_r is the (unique) minimizer to the problem

$$\min_{d \in \mathbb{R}^n} \frac{1}{2}\rho\|d\|^2 + \hat{f}_r(y + d), \tag{4}$$

whose objective function is obtained by adding the proximity term $\frac{1}{2}\rho\|d\|^2$ to the partial cutting-plane function $\hat{f}_r$. We remind a well-known fact about bundle methods, that the introduction of a proximity term into the objective function is aimed both at stabilizing the cutting plane approach and at getting a well-posed problem, independently of possible unboundedness of the partial cutting-plane function. In the following the scalar parameter ρ will be referred to as the *proximity parameter*.

By introducing the additional (scalar) variable v, problem (4) can be rewritten as the following convex quadratic program in $\mathbb{R}^{n+1}$

$$QP(\mathcal{B}_r) \quad \min_{v,d} \quad \frac{1}{2}\rho\|d\|^2 + v, \quad v \ge g_{i_k}^\top d + \beta_k \quad \forall k \in \mathcal{I}_r.$$

Letting (v_r, d_r) be the optimal solution to $QP(\mathcal{B}_r)$ we have, of course, that

$$v_r = \hat{f}_r(y + d_r).$$

The dual of problem $QP(\mathcal{B}_r)$ has the form:

$$DP(\mathcal{B}_r) \quad \min_{\lambda \in \mathbb{R}^r} \quad \frac{1}{2\rho} \left\| \sum_{k \in \mathcal{I}_r} \lambda_k g_{i_k} \right\|^2 - \sum_{k \in \mathcal{I}_r} \lambda_k \beta_k,$$
$$\sum_{k \in \mathcal{I}_r} \lambda_k = 1,$$
$$\lambda_k \geq 0 \quad \forall k \in \mathcal{I}_r.$$

By letting $\lambda_k^{(r)}$, for every $k \in \mathcal{I}_r$, be the optimal solution to $DP(\mathcal{B}_r)$, the following relations, linking primal and dual optimal variables, hold:

$$d_r = -\frac{1}{\rho} g^{(r)}, \tag{5a}$$

$$v_r = -\frac{1}{\rho} \|g^{(r)}\|^2 + \sum_{k \in \mathcal{I}_r} \lambda_k^{(r)} \beta_k, \tag{5b}$$

where $g^{(r)} \stackrel{\triangle}{=} \sum_{k \in \mathcal{I}_r} \lambda_k^{(r)} g_{i_k}$. We remark that some useful information can be obtained from the solution of $DP(\mathcal{B}_r)$. In fact, as proved in [6, Lemma 2.2], the vector $g^{(r)}$ is an ϵ-subgradient of f at y, i.e.,

$$g^{(r)} \in \partial_\epsilon f(y), \tag{6}$$

where

$$\epsilon \stackrel{\triangle}{=} f(y) - \sum_{k \in \mathcal{I}_r} \lambda_k^{(r)} \beta_k \geq 0.$$

Such a property allows to establish an approximate optimality condition. In fact whenever both $\|g^{(r)}\|$ and ϵ are "small," we accept y as an approximate optimal solution.

In standard cutting-plane approaches every time a new iterate is generated, the actual value of the objective function at such a point is not smaller than the *predicted* one, i.e., than the cutting-plane function value. This in turn implies that the newly generated affine piece cuts part of the current cutting-plane function epigraph, and consequently a tighter model of the objective function is obtained.

This is no longer the case in the algorithmic framework presented in [6], where any new affine piece is not obtained, in general, by evaluating the actual objective function value (which would require calculation of the maximal component function), but by evaluating just one component function, which might not result greater than the corresponding value of the partial cutting-plane function. In such case no cut would take place and consequently no useful information would be extracted from the newly generated point. In this respect the following two properties play a significant role.

Decrease property (see [6, Lemma 2.3]) Let δ be any positive scalar. If $f_i(y+d_r) \leq v_r + \delta$ for every $i \in I$, then

$$f(y+d_r) - f(y) \leq -\frac{1}{\rho}\|g^{(r)}\|^2 + \delta.$$

Cut property (see [6, Lemma 2.4]) Let δ be any positive scalar. If $f_h(y+d_r) > v_r + \delta$ for some index $h \in I$, then (v_r, d_r) is not feasible for problem $QP(\mathcal{B}_{r+1})$ obtained by modifying the bundle so that

$$\mathcal{B}_{r+1} = \mathcal{B}_r \cup \{(x_{r+1},\, f_h(x_{r+1}),\, g_h,\, \beta_{r+1})\},$$

where $x_{r+1} = y + d_r$, $g_h \in \partial f_h(x_{r+1})$, and $\beta_{r+1} = f_h(x_{r+1}) + g_h^\top(y - x_{r+1})$.

The meaning of the two properties is the following. First notice that whenever a component $i \in I$ is selected so that at d_r the inequality

$$f_i(y+d_r) > v_r + \delta$$

is satisfied, then a significant cut is obtained (i.e., *cut property* holds). Moreover, if no such cut is obtained for all the component functions, then a sufficient reduction of the objective function in passing from y to $y + d_r$ is guaranteed (i.e., *decrease property* holds), irrespective of whether or not the actual value of f at y is known, provided that an appropriate choice of some parameters is made (see (7) and Algorithm 2 for details).

The above observations drive the method described below. In the algorithm the point $y + d_r$ is accepted as a *bundle enrichment* whenever an index satisfying the cut property is found, whilst if no such index exists, (i.e., the decrease property is satisfied), then the point $y + d_r$ becomes the new stability center. Observe, of course, that whenever the latter case occurs an exact calculation of the objective function will actually take place.

The incremental algorithm is based on repeatedly solving problems of the type $QP(\mathcal{B}_r)$, or equivalently $DP(\mathcal{B}_r)$, and on the evaluation of just one component function at a time.

The initialization of the algorithm requires a starting point $x_1 \in \mathbb{R}^n$. The initial stability center y is set equal to x_1, since we assume that the first bundle element always denotes the current stability center. Thus, the initial bundle $\mathcal{B}_1$ is made up of just one element $(y, f_{i_1}(y), g_{i_1}(y), \beta_1)$, for some $i_1 \in I$, where $g_{i_1}(y) \in \partial f_{i_1}(y)$, with $g_{i_1}(y) \neq 0$, and $\beta_1 = f_{i_1}(y)$. Of course, if such an index i_1 does not exist (i.e., $g_i(y) = 0\ \forall i \in I$) then the starting point x_1 is optimal.

The following global parameters are to be set:

- the optimality tolerance $\eta > 0$ and the approximation measure $\epsilon > 0$;
- the increase parameter $\sigma > 1$.

In addition, two positive parameters subject to possible modifications are needed, the descent parameter δ and the proximity parameter ρ; their initial setting is, respectively, $\delta := \bar{\delta} > 0$ and $\rho := \bar{\rho} > 0$. It is also assumed that parameters η, $\bar{\delta}$ and $\bar{\rho}$ satisfy

the condition:

$$\eta^2 > \bar{\rho}\bar{\delta}. \tag{7}$$

The algorithm can be summarized as follows.

Algorithm 1 (Algorithm Outline)

1. Initialize.
2. Execute the "main iteration."
3. Update the bundle with respect to the new stability center and return to 2.

In the sequel we describe in details the "main iteration," i.e., the sequence of steps where the stability center remains unchanged.

Possible exits from the "main iteration" are:

(i) termination of the whole algorithm, resulting from the satisfaction of an approximate optimality condition;
(ii) update of the stability center, resulting from the satisfaction of the decrease property.

For sake of notation simplicity the "main iteration" is not indexed (e.g., the bundle $\mathcal{B}_r$ and the bundle index set $\mathcal{I}_r$ will be indicated by $\mathcal{B}$ and $\mathcal{I}$, respectively). Of course the stability center y is to be intended as the current stability center.

The "main iteration" maintains the (updated) bundle of information from previous iterations. Updating the bundle is necessary since the β_ks are dependent on the stability center.

Algorithm 2 (Main Iteration)

1. Solve $QP(\mathcal{B})$ or, alternatively, $DP(\mathcal{B})$, and obtain both the optimal primal solution $(\bar{v}, \bar{d})$ and the optimal dual solution $\bar{\lambda}$. Set $\bar{x} = y + \bar{d}$.
   If
   $$\left\| \sum_{k \in \mathcal{I}} \bar{\lambda}_k g_{i_k} \right\| > \eta$$
   then go to Step 3.
2. Calculate $f(y)$. If
   $$f(y) - \sum_{k \in \mathcal{I}} \bar{\lambda}_k \beta_k \leq \epsilon$$
   then STOP (approximate optimality achieved).
   Else set $\rho := \sigma\rho$, $\delta := \delta/\sigma$, make a bundle reset, i.e. set
   $$\mathcal{B} := \{(y, f(y), g(y), \beta_1)\},$$
   and
   $$\mathcal{I} := \{1\},$$
   where $g(y) \in \partial f(y)$ and $\beta_1 = f(y)$, and return to Step 1.

3. Extract any index i from the function index set I. If

$$f_i(\bar{x}) > \bar{v} + \delta$$

then (cut property satisfied) set

$$\mathcal{B} := \mathcal{B} \cup \{(\bar{x}, f_i(\bar{x}), g_i, \beta_i\}$$

where $g_i \in \partial f_i(\bar{x})$ and $\beta_i = f_i(\bar{x}) - g_i^\top \bar{d}$. Update appropriately the bundle index set $\mathcal{I}$, restore the function index set by setting $I = \{1, \ldots, m\}$ and return to Step 1.

4. Set $I = I - \{i\}$. If

$$I = \emptyset$$

then (decrease property satisfied) set $y = \bar{x}$ and EXIT (update the stability center). Else return to Step 3.

Some explanations are in order. The aim of Step 2 is to check satisfaction of the approximate optimality condition derived from (6). If this is the case, then successful termination of the whole algorithm takes place; indeed, we have $\| \sum_{k \in \mathcal{I}} \bar{\lambda}_k g_{i_k} \| \leq \eta$ and $f(y) - \sum_{k \in \mathcal{I}} \bar{\lambda}_k \beta_k \leq \epsilon$, which, taking into account (6), imply that the current stability center y fulfills the following approximate optimality condition:

$$f(x) \geq f(y) - \eta \|x - y\| - \epsilon \quad \forall x \in \mathbb{R}^n.$$

If termination does not occur, a suitable update of the parameters ρ and δ, along with a reset of the bundle, must be executed before solving again $QP(\mathcal{B})$ or $DP(\mathcal{B})$. In this respect, we point out that the condition (7), holding for the initial setting of the parameters η, ρ, and δ, is valid throughout the algorithm, as a consequence of the simultaneous increase of ρ and reduction of δ which takes place every time the parameter updates at Step 2 occur.

It is also worth noting that every update of the stability center occurs at Step 4 whenever the decrease property is satisfied, i.e., all the component functions have been evaluated. This implies that all the stability centers, but possibly the first, are inserted into the bundle carrying the information related to the actual value of the objective function f at y. Hence, the explicit evaluation of $f(y)$, requested at Step 2, is just a safeguard against the possible use of inexact information about y the first time Step 2 is entered, that is, when the stability center has never been updated.

We refer the interested reader to [6] for a complete discussion on the convergence properties of the algorithm.

3 Incremental min-max and Lagrangian relaxation

Consider the following integer programming problem

$$\begin{aligned} IP \quad z_{IP} = \max \quad & c^\top x, \\ & Ax \le b, \\ & Dx = e, \\ & x \in \mathbb{Z}^n_+, \end{aligned}$$

where $A \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^p$, $e \in \mathbb{R}^m$, and $\mathbb{Z}^n_+$ denotes the set of n-dimensional vectors of nonnegative integers. We suppose, as usual in the Lagrangian relaxation approach [4, 7], that a set of complicating constraints can be identified in *IP*. In particular, for reasons that will become clear later, we assume that the equality constraints $Dx = e$ are the complicating ones. Lagrangian relaxation of *IP*, dependent on the unconstrained multiplier vector $\mu \in \mathbb{R}^m$, is then the following problem:

$$\begin{aligned} LR(\mu) \quad z(\mu) = \max_x \quad & c^\top x + \mu^\top (e - Dx), \\ & Ax \le b, \\ & x \in \mathbb{Z}^n_+. \end{aligned}$$

The optimal values of *IP* and $LR(\mu)$ satisfy the condition $z(\mu) \ge z_{IP}$, for every $\mu \in \mathbb{R}^m$, i.e., the optimal objective function value of the Lagrangian relaxation is an upper bound on the optimal objective function value of the original problem. The best (minimal) upper bound can then be calculated by solving the so-called Lagrangian dual problem:

$$z_{LD} = \min_\mu z(\mu).$$

If we assume that $LR(\mu)$ has finitely many feasible solutions, we can in principle enumerate them as $x^{(1)}, x^{(2)}, \ldots, x^{(s)}$, namely

$$\{x^{(1)}, \ldots, x^{(s)}\} = \{x \in \mathbb{Z}^n_+ : Ax \le b\},$$

and rewrite its objective function in the following way

$$z(\mu) = \max_{1\le t\le s} \{\ell_t + \mu^\top g_t\},$$

where $\ell_t \triangleq c^\top x^{(t)}$ and $g_t \triangleq e - Dx^{(t)}$. Hence, the Lagrangian dual problem can be stated as

$$LD \quad z_{LD} = \min_{\mu\in\mathbb{R}^m} \max_{1\le t\le s} \{\ell_t + \mu^\top g_t\},$$

and actually consists in a finite min-max problem, where the objective function is a piecewise affine convex function of the max type, defined as the pointwise maximum of a possibly fairly large set of component functions, each of them being affine.

We can now recognize that the Lagrangian dual formulation *LD* looks like a convex finite min-max problem equivalent to problem (1), and therefore it can be tackled, mutatis mutandis, by means of the incremental approach presented in Sect. 2. We remark that two assumptions introduced earlier play a fundamental role in turning the Lagrangian dual into an unconstrained convex finite min-max problem. In fact, on one hand, assuming that equality constraints are the complicating ones, i.e., constraints to relax, allows to have unconstrained multipliers. On the other hand, assuming that $LR(\mu)$ has finitely many feasible solutions guarantees that the objective function of *LD* be a finite max-function.

The latter observation raises the natural question about whether one actually needs to know every feasible solution of $LR(\mu)$. We notice that the affine functions $\ell_t + \mu^\top g_t, t = 1, \ldots, s$, play the role of the max-function components, and hence any feasible solution of the Lagrangian relaxation generates a component function. Therefore, facing the minimization of $z(\mu)$ via a naive application of Algorithm 1 would imply the need for possibly evaluating all the feasible solutions of $LR(\mu)$ for a given μ, quite an unrealistic assumption for any significant instance of an integer program. This is the reason why we need to design an appropriate version of the main iteration (Algorithm 2), which can take into account the different context of application, while maintaining active the underlying incremental philosophy. In particular, our method turns out to be particularly suitable for the rather common case when the Lagrangian relaxation is not easy to solve, but there exist effective heuristic techniques to tackle it.

Suppose we are given a certain set of points $\mu_k \in \mathbb{R}^m$, $k = 1, \ldots, r$, and let us focus on the particularization of the bundle. Let also y be the current stability center, possibly coinciding with one of the μ_k. Following the definitions of Sect. 2, the bundle has the form

$$\mathcal{B}_r = \{(\mu_k, z_{t_k}, g_{t_k}, \beta_{t_k}), \forall k \in \mathcal{I}_r\},$$

where the index set $\mathcal{I}_r \stackrel{\triangle}{=} \{1, \ldots, r\}$ is the *bundle index set*; $z_{t_k} = \ell_{t_k} + \mu_k^\top g_{t_k}$, for some $t_k \in \{1, \ldots, s\}$, identifies the component function actually chosen at point μ_k corresponding to some feasible solution $x^{(t_k)}$; $g_{t_k} = e - Dx^{(t_k)}$, and β_{t_k}, following (3), has the form

$$\beta_{t_k} = \ell_{t_k} + g_{t_k}^\top \mu_k + g_{t_k}^\top (y - \mu_k) = \ell_{t_k} + g_{t_k}^\top y.$$

Assuming that the bundle $\mathcal{B}_r$ is currently available, we calculate a new point μ_{r+1} by setting

$$\mu_{r+1} = \mu_r + d_r,$$

where d_r is the (unique) minimizer to the quadratic problem of the form $QP(\mathcal{B}_r)$ defined in Sect. 2. By letting $\lambda_k^{(r)}$, for every $k \in \mathcal{I}_r$, be the optimal solution of the corresponding dual problem $DP(\mathcal{B}_r)$, relations (5) linking primal and dual optimal variables still hold.

Now we are ready to restate Algorithm 1; more precisely, since the initialization phase does not change, we will just focus on restating its main iteration. Speaking of which, we remark that in the remainder of the paper, slightly abusing of our terminology, we will often refer to a given main iteration as an algorithm for solving an

associated min-max problem (e.g., problem *LD*), with the obvious meaning that Algorithm 1 actually solves that problem, once the appropriate main iteration has been embedded into it.

Some preliminary remarks are useful to understand the structure of the main iteration. In Sect. 2, the assumption underlying Algorithm 2 was the availability of all its component functions; we just aimed at avoiding to evaluate all of them at every point. In the Lagrangian dual case, we simply do not know every feasible point of the Lagrangian relaxation, hence we cannot evaluate all the component functions at a given point. What we can do instead is to partially enumerate the feasible solutions, as long as either the cut property is satisfied, and the bundle is updated, or such an enumeration process cannot carry on, and obviously the Lagrangian relaxation needs to be solved via an exact approach. As for the enumeration process, several alternatives are available. On the one hand, assuming that h different heuristic algorithms ($h \geq 1$) are available to find a feasible solution to the Lagrangian relaxation, we could generate feasible solutions trying each of such heuristics until a cut is generated, eventually resorting to an exact method when no more heuristics are available. On the other hand, the sequence of feasible solutions could even be generated directly via an exact method, e.g., a branch-and-bound approach, by simply checking the cut property every time the best integer solution is updated; in this case the enumeration process, if no cut is ever generated, would naturally terminate returning the exact solution of the Lagrangian relaxation. Of course, one can easily understand that any hybrid approach between the two listed above, would work similarly. Moreover, depending on the heuristic algorithms available for a given problem, the enumeration process could even become much more structured than the one just described.

Once recognized that several choices for generating a sequence of feasible solutions for $LR(\mu)$ are available, we state the main iteration for problem *LD* adopting a heuristic-based enumeration process. Hence, let $H = \{1, \ldots, h\}$ be the *heuristics index set*, and assume that an exact algorithm to solve $LR(\mu)$ is available. Again for sake of notation simplicity we do not index the "main iteration."

Algorithm 3 (Main Iteration)

1. Solve $QP(\mathcal{B})$ or, alternatively, $DP(\mathcal{B})$, and obtain both the optimal primal solution $(\bar{v}, \bar{d})$ and the optimal dual solution $\bar{\lambda}$. Set $\bar{\mu} = y + \bar{d}$.
   If
   $$\left\| \sum_{k \in \mathcal{I}} \bar{\lambda}_k g_{t_k} \right\| > \eta$$
   then go to Step 3.
2. Calculate $z(y)$ by finding the optimal solution $x^\star$ to the problem $LR(y)$. If
   $$z(y) - \sum_{k \in \mathcal{I}} \bar{\lambda}_k \beta_k \leq \epsilon$$
   then STOP (approximate optimality achieved).
   Else set $\rho := \sigma\rho$, $\delta := \delta/\sigma$, make a bundle reset, i.e., set
   $$\mathcal{B} := \{(y, z(y), g(y), \beta_1)\},$$

and

$$\mathcal{I} := \{1\},$$

where $g(y) = e - Dx^\star$ and $\beta_1 = z(y)$, and return to Step 1.

3. Find a feasible solution $x^{(t)}$ to the problem $LR(\bar{\mu})$. If

$$z_t(\bar{\mu}) > \bar{v} + \delta \tag{8}$$

then (cut property satisfied) set

$$\mathcal{B} := \mathcal{B} \cup \{(\bar{\mu}, z_t(\bar{\mu}), g_t, \beta_t\},$$

where $g_t = e - Dx^{(t)}$ and $\beta_t = c^\top x^{(t)} + g_t^\top \bar{\mu} - g_t^\top \bar{d}$. Update appropriately the bundle index set $\mathcal{I}$ and return to Step 1.

4. Set $H = H - \{t\}$. If

$$H = \emptyset$$

(no more heuristics available) then find the exact solution to the problem $LR(\bar{\mu})$ (decrease property satisfied), set $y = \bar{\mu}$ and EXIT (update of the stability center). Else return to Step 3.

In Sect. 4 we will give practical details on how to exploit Algorithm 3 for solving the Lagrangian dual of GAP.

4 An incremental Lagrangian dual approach for GAP

The Generalized Assignment Problem consists in finding the optimal assignment of a given number of *tasks* to a set of *agents*, under the constraints that each task be assigned to exactly one agent, and the total consumption of a given resource cannot exceed, for each agent, the available amount. The objective is in general put in the form of maximization of the assignment value. Suppose, in particular, that a set of tasks $\mathcal{J} = \{1, \ldots, n\}$ is given, together with a set of agents $\mathcal{I} = \{1, \ldots, m\}$. Let c_{ij} be the value of assigning task j to agent i, a_{ij} be the amount of resource consumed by task j when assigned to agent i, and b_i be the resource availability of agent i. We assume that c_{ij}, a_{ij}, and b_i are all nonnegative scalars. The decision variables of the problem are the binary variables x_{ij}, $i \in \mathcal{I}$ and $j \in \mathcal{J}$, defined as:

$$x_{ij} = \begin{cases} 1 & \text{if task } j \text{ is assigned to agent } i, \\ 0 & \text{otherwise.} \end{cases}$$

GAP is then formulated as:

$$z = \max \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} x_{ij}, \tag{9a}$$

$$\sum_{i \in \mathcal{I}} x_{ij} = 1 \quad \forall j \in \mathcal{J}, \tag{9b}$$

$$\sum_{j\in\mathcal{J}} a_{ij}x_{ij} \le b_i \quad \forall i \in \mathcal{I}, \tag{9c}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \tag{9d}$$

where constraint (9b) are called semi-assignment constraints, whereas constraints (9c) are the capacity (or knapsack) constraints.

The Lagrangian relaxation obtained dualizing the semi-assignment constraints (9b) by means of unconstrained multipliers μ_j, $j \in \mathcal{J}$, is:

$$z(\mu) = \max \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}x_{ij} + \sum_{j\in\mathcal{J}} \mu_j \left(1 - \sum_{i\in\mathcal{I}} x_{ij}\right),$$

$$\sum_{j\in\mathcal{J}} a_{ij}x_{ij} \le b_i \quad \forall i \in \mathcal{I},$$

$$x_{ij} \in \{0,1\} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}.$$

It is a decomposable problem which can be written in the form:

$$z(\mu) = \sum_{j\in\mathcal{J}} \mu_j + \sum_{i\in\mathcal{I}} Z_i(\mu),$$

where

$$Z_i(\mu) \stackrel{\triangle}{=} \max \sum_{j\in\mathcal{J}} (c_{ij} - \mu_j)x_{ij},$$

$$\sum_{j\in\mathcal{J}} a_{ij}x_{ij} \le b_i,$$

$$x_{ij} \in \{0,1\} \quad \forall j \in \mathcal{J}.$$

It is easy to recognize that calculation of $Z_i(\mu)$ requires solution of a binary knapsack problem (note that at the optimum there holds $x_{ij} = 0$ if $c_{ij} - \mu_j \le 0$). As a consequence, solving the Lagrangian relaxation of GAP, for a given multiplier vector μ, amounts to solving m binary knapsack problems.

The Lagrangian dual consists in finding the best upper bound to the optimal value of GAP, i.e., in solving

$$\min_{\mu\in\mathbb{R}^n} z(\mu) = \min_{\mu\in\mathbb{R}^n} \sum_{j\in\mathcal{J}} \mu_j + \sum_{i\in\mathcal{I}} Z_i(\mu).$$

Observe that such objective function is in the form of sum of a linear term plus the sum of m functions of the max type, and that the evaluation of each of the latter ones requires solution to a binary knapsack problem.

The Lagrangian dual can be easily rewritten in a form suitable for application of the incremental algorithm described in Sect. 3. In fact, since the number of feasible

solutions of the Lagrangian relaxation is finite, by enumerating them as

$$x_{ij}^{(1)}, x_{ij}^{(2)}, \ldots, x_{ij}^{(s)} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J},$$

we can rewrite the Lagrangian dual in the form:

$$\min_{\mu} z(\mu) = \min_{\mu} \max_{t \in S} \left\{ \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} x_{ij}^{(t)} + \sum_{j \in \mathcal{J}} \mu_j g_j^{(t)} \right\}, \tag{10}$$

where $S \triangleq \{1, \ldots, s\}$, and

$$g_j^{(t)} = \left(1 - \sum_{i \in \mathcal{I}} x_{ij}^{(t)}\right).$$

The formulation (10) emphasizes the fact that $z(\mu)$ is the pointwise maximum of s affine functions in the variable μ. Of course the vector $g^{(t)} = (g_1^{(t)}, g_2^{(t)}, \ldots, g_n^{(t)})^\top$ is the gradient of the t-th affine piece, for any $t \in S$.

The minimization of $z(\mu)$ can be tackled by adopting different nonsmooth descent techniques like subgradient type methods or any cutting-plane based approach. In all such cases indeed it is required the generation of a sequence of points $\mu^{(k)}$, $k = 1, 2, \ldots$ where the function $z(\mu^{(k)})$ is to be evaluated exactly, together with a subgradient $g^{(k)}$. Calculation of $z(\mu^{(k)})$ requires exact solution of m knapsack problems, i.e., implicit enumeration of all feasible solutions $x_{ij}^{(t)}$, $t \in S$, $i \in \mathcal{I}$, $j \in \mathcal{J}$. On the other hand, the adoption of the incremental approach, described in Sect. 3, would require only to pick up one component function, i.e.,

$$\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} x_{ij}^{(t_k)} + \sum_{j \in \mathcal{J}} \mu_j^{(k)} g_j^{(t_k)},$$

with $t_k \in S$, which in turn requires only to determine one feasible solution $x_{ij}^{(t_k)}$, $i \in \mathcal{I}$, $j \in \mathcal{J}$.

The above observation suggests the opportunity of replacing the optimal solution of the m knapsack problems by any suboptimal solution provided by appropriate heuristic algorithms. Of course, we note that, in running Algorithm 3, it might well happen that, because of possible failure of the cut property, at least one of the m knapsack problems is to be solved exactly.

Summarizing, if a set of points $\mu^{(k)} \in \mathbb{R}^n$, $k = 1, \ldots, r$, is given along with the current stability center y, in the GAP case, we can define the "bundle" $\mathcal{B}_r$ as the following set of elements:

$$\mathcal{B}_r \triangleq \{(\mu^{(k)}, z_{t_k}, g^{(t_k)}, \beta_{t_k}), \forall k \in \mathcal{I}_r\},$$

where the index set $\mathcal{I}_r \triangleq \{1, \ldots, r\}$ is the *bundle index set*, $g^{(t_k)} = (g_1^{(t_k)}, \ldots, g_n^{(t_k)})^\top$,

$$z_{t_k} = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} x_{ij}^{(t_k)} + \sum_{j \in \mathcal{J}} \mu_j^{(k)} g_j^{(t_k)},$$

and

$$\beta_{t_k} = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} x_{ij}^{(t_k)} + \sum_{j \in \mathcal{J}} y_j g_j^{(t_k)}$$

with y_j denoting the j-th component of vector y. With the above definitions, Algorithm 3 (embedded into Algorithm 1) is an incremental cutting plane method for solving the Lagrangian dual of GAP. The initialization of the algorithm requires a starting point $\mu^{(1)} \in \mathbb{R}^n$ that can be obtained by using the initialization phase of the multiplier adjustment method described in [5]; hence, we set

$$\mu_j^{(1)} = \max_2 \{c_{ij} : i \in I\} \quad \forall j \in J,$$

namely, each component $\mu_j^{(1)}$ is the second maximum over all the coefficients c_{ij}.

5 Numerical results

We have tested Algorithm 3, appropriately tailored for GAP, on 90 test problems grouped in the literature in two sets: the first 60 problems, `gap1-1` to `gap12-5`, are classified as "small-sized," while the remaining 30 problems are classified as "large-sized," and referred to as types A, B, C, D, and E. These test problems are taken from the OR-Library maintained by J.E. Beasley [1], and available on the web at http://people.brunel.ac.uk/~mastjjb/jeb/orlib/gapinfo.html.

The algorithm, called IGAP, has been implemented in Fortran 90 on a Pentium IV −1 GHz personal computer running on a Windows XP system. The following input parameters have been used: $\epsilon = 0.001$, $\eta = 0.05$, $\bar{\delta} = 0.08$, $\bar{\rho} = 0.009$ and $\sigma = 10$. Some exceptions occur due to failed termination of the quadratic programming solver, i.e., the subroutine DQPROG provided by the IMSL library and based on M.J.D. Powell's implementation of the Goldfarb and Idnani [8] dual quadratic programming algorithm. Thus, problem instances highlighted with $*$ and $**$, have been solved by setting $\bar{\rho} = 0.01$ and $\bar{\rho} = 0.03$, respectively. IGAP embeds the subroutine MT1R as an exact solver for the knapsack problem; MT1R is a branch and bound algorithm devised and implemented by Martello and Toth, and described in [11, Sect. 2.5.2]. To emphasize the role played by the incremental philosophy we compare our method with a fairly similar algorithm which is based on a plain implementation of the bundle method philosophy [10]. More precisely, our implementation of the bundle algorithm consists in applying IGAP by enforcing exact evaluation of the objective function whenever a new multiplier vector is generated. In Tables 1 and 2, for each instance we report the optimal value z_{IP} of GAP, the objective function value $\bar{z}_{LD}$ of the Lagrangian dual returned by the algorithm upon termination, the corresponding number N_{ex} of exact objective function evaluations, and the elapsed CPU times. In addition, for the incremental case we provide the number of times N_{heu} a heuristic solution to the Lagrangian relaxation has been calculated. In our implementation, the greedy algorithm described in [11] has been adopted. The greedy algorithm sorts the items according to decreasing values of the profit per unit weight,

Table 1 IGAP: small-sized instances

Type	Characteristics			Bundle			IGAP			
	m	n	z_{IP}	N_{ex}	$\bar{z}_{LD}$	Time	N_{ex}	N_{heu}	$\bar{z}_{LD}$	Time
gap1-1*	5	15	336	71	337.00	0.59	36	156	337.00	2.06
gap1-2	5	15	327	87	327.00	0.64	43	114	327.00	0.44
gap1-3*	5	15	339	90	339.50	0.98	25	115	339.50	0.36
gap1-4	5	15	341	63	341.00	0.17	75	142	341.00	0.08
gap1-5	5	15	326	85	327.25	0.58	71	179	327.25	0.78
gap2-1	5	20	434	127	435.00	2.72	43	153	435.00	2.27
gap2-2	5	20	436	145	436.00	3.53	41	174	436.00	1.22
gap2-3	5	20	420	113	420.80	2.33	63	186	420.75	1.30
gap2-4	5	20	419	133	420.00	2.69	114	247	419.50	3.67
gap2-5	5	20	428	107	428.00	2.55	39	176	428.00	1.00
gap3-1	5	25	580	160	580.00	5.16	72	238	580.00	3.06
gap3-2	5	25	564	162	564.00	4.53	47	206	564.00	2.63
gap3-3	5	25	573	145	573.00	4.42	36	161	573.00	1.28
gap3-4	5	25	570	163	570.00	5.19	31	180	570.00	1.75
gap3-5	5	25	564	193	564.14	6.38	67	230	564.00	3.11
gap4-1	5	30	656	229	656.86	9.36	136	282	656.78	4.11
gap4-2*	5	30	644	224	646.45	8.36	67	253	646.45	5.64
gap4-3	5	30	673	287	674.50	13.72	59	252	674.39	5.66
gap4-4	5	30	647	251	647.60	10.48	71	286	647.50	6.83
gap4-5	5	30	664	195	664.00	7.59	37	230	664.00	4.23
gap5-1	8	24	563	183	564.06	5.81	56	225	564.00	4.23
gap5-2	8	24	558	190	558.00	6.97	86	276	558.00	4.69
gap5-3	8	24	564	156	564.00	4.92	33	208	564.00	3.42
gap5-4	8	24	568	162	568.71	4.42	103	293	568.71	6.17
gap5-5	8	24	559	205	560.60	6.92	88	264	560.57	4.61
gap6-1	8	32	761	205	762.10	6.63	74	275	762.10	4.67
gap6-2*	8	32	759	210	760.00	8.13	72	322	760.00	9.00
gap6-3	8	32	758	235	758.50	9.94	68	307	758.50	8.84
gap6-4	8	32	752	261	753.00	10.50	81	260	753.00	4.95
gap6-5	8	32	747	314	747.80	13.06	77	319	747.80	6.77
gap7-1	8	40	942	407	943.06	19.58	170	443	943.07	13.38
gap7-2	8	40	949	433	949.46	21.86	69	370	949.43	10.13
gap7-3	8	40	968	346	968.00	15.94	51	443	968.00	14.92
gap7-4	8	40	945	445	945.00	21.58	93	517	945.00	20.05
gap7-5	8	40	951	308	952.00	13.48	103	342	952.00	8.22
gap8-1*	8	48	1133	542	1133.97	27.73	215	669	1133.54	24.06
gap8-2	8	48	1134	337	1135.50	14.52	119	471	1135.72	15.55

Table 1 (*Continued*)

Type	Characteristics			Bundle			IGAP			
	m	n	z_{IP}	N_{ex}	$\bar{z}_{LD}$	Time	N_{ex}	N_{heu}	$\bar{z}_{LD}$	Time
gap8-3	8	48	1141	596	1141.00	30.45	145	452	1141.33	17.50
gap8-4	8	48	1117	635	1118.53	33.11	225	782	1118.50	29.59
gap8-5*	8	48	1127	608	1127.15	31.84	243	671	1127.00	26.58
gap9-1	10	30	709	184	710.00	6.48	39	273	710.00	7.03
gap9-2	10	30	717	270	717.33	10.27	105	318	717.00	7.36
gap9-3	10	30	712	177	713.00	5.94	57	227	713.00	4.44
gap9-4	10	30	723	260	724.00	10.36	42	303	724.00	7.50
gap9-5	10	30	706	190	707.50	6.91	97	278	708.00	4.64
gap10-1	10	40	958	392	958.00	18.45	56	489	958.00	16.38
gap10-2	10	40	963	258	964.00	10.73	104	406	964.00	11.30
gap10-3	10	40	960	333	960.23	14.06	98	411	960.15	14.59
gap10-4	10	40	947	325	947.00	13.97	129	511	947.00	17.02
gap10-5	10	40	947	364	948.35	16.80	161	507	948.25	16.70
gap11-1*	10	50	1139	613	1139.50	33.92	153	490	1139.42	17.30
gap11-2	10	50	1178	536	1178.33	30.11	212	519	1178.33	18.75
gap11-3*	10	50	1195	420	1195.17	20.94	115	570	1195.17	16.00
gap11-4**	10	50	1171	265	1172.00	12.64	105	400	1172.00	12.41
gap11-5*	10	50	1171	561	1172.50	31.75	210	558	1172.39	22.16
gap12-1	10	60	1451	590	1451.00	28.73	99	668	1451.00	24.45
gap12-2*	10	60	1449	931	1449.88	54.98	185	630	1449.88	23.31
gap12-3	10	60	1433	583	1433.50	29.34	83	556	1433.50	21.95
gap12-4	10	60	1447	699	1447.71	38.47	91	715	1447.60	26.52
gap12-5*	10	60	1446	463	1446.50	21.05	312	818	1446.50	33.42

and then proceeds to insert them into the knapsack, starting from the first element and until there is no more space available in the sack.

It is worth noting that, unlike the GAP formulation given in Sect. 4, instances A–E are available as minimization problems and, therefore, the results are presented accordingly. In particular, this means that for such instances the values $\bar{z}_{LD}$ are lower bounds for z_{IP}. Tables 1 and 2 show that for 89 instances out of 90 the IGAP algorithm behaves better than the standard bundle in terms of the number of exact objective function evaluations, which is very often at least halved in the incremental case. The counterpart is the necessity of calculating a certain number of heuristic solutions to the Lagrangian relaxation. This has indeed some influence on computation times, where IGAP outperforms the standard bundle on 58 instances out of 90. Such a behavior can be easily explained by observing that the computational burden of solving exactly the knapsack problem need not be much heavier than applying a heuristic algorithm to obtain a feasible solution. Of course, we expect that using the incremental approach to solve Lagrangian dual becomes more and more effective for

Table 2 IGAP: type A–E instances

Type	Characteristics			Bundle			IGAP			
	m	n	z_{IP}	N_{ex}	$\bar{z}_{LD}$	Time	N_{ex}	N_{heu}	$\bar{z}_{LD}$	Time
A	5	100	1698	99	1697.48	0.03	28	100	1697.55	0.02
A	5	200	3235	125	3233.68	0.05	47	146	3233.93	0.03
A	10	100	1360	110	1357.34	0.05	44	138	1358.52	0.05
A	10	200	2623	157	2622.94	0.06	61	178	2622.40	0.08
A	20	100	1158	142	1157.14	0.06	26	144	1157.17	0.06
A	20	200	2339	191	2236.79	0.08	28	175	2337.21	0.69
B	5	100	1843	229	1831.66	0.05	72	267	1830.64	0.08
B	5	200	3552	344	3545.78	0.05	119	422	3545.39	0.19
B	10	100	1407	174	1404.63	0.09	77	185	1404.38	0.08
B	10	200	2827	362	2822.15	0.14	212	480	2820.89	0.22
B	20	100	1166	220	1165.01	0.06	98	238	1165.14	0.36
B	20	200	2339	269	2337.38	0.20	147	330	2337.20	1.30
C	5	100	1931	162	1917.05	0.08	98	275	1922.59	0.11
C	5	200	3456	325	3448.77	0.08	147	426	3449.13	0.14
C	10	100	1402	202	1392.81	0.13	109	278	1394.44	0.08
C	10	200	2806	314	2799.91	0.06	115	431	2801.41	0.11
C	20	100	1243	260	1239.56	0.05	138	283	1238.79	0.30
C	20	200	2391	429	2388.10	0.19	143	474	2387.78	1.89
D	5	100	6353	365	6347.49	0.06	108	403	6347.45	0.09
D	5	200	12742	433	12738.67	0.23	128	607	12738.96	0.16
D	10	100	6348	395	6339.51	0.09	137	408	6339.44	0.14
D	10	200	12432	572	12311.94	7.63	175	612	12311.52	0.39
D	20	100	6190	464	6174.71	0.34	226	559	6174.82	0.72
D	20	200	12241	707	12141.19	8.20	269	823	12142.07	5.45
E	5	100	12681	551	12657.73	0.09	353	643	12657.87	0.14
E	5	200	24930	808	24896.21	0.08	477	985	24902.04	0.08
E	10	100	11577	633	11533.36	0.13	573	796	11536.96	0.11
E	10	200	23307	1028	23281.01	0.22	916	1334	23286.62	0.19
E	20	100	8436	801	8407.66	0.23	634	824	8405.14	0.33
E	20	200	22379	1600	22349.08	8.84	1424	1758	22350.25	12.78

those problems for which heuristic algorithms are available that have much lower computational costs than their exact counterpart.

As mentioned in the introduction, very often in implementing enumeration techniques, multiplier adjustment is employed at the nodes of the enumeration tree to possibly improve the quality of the bound. Extensive use of subgradient techniques is made in such context, for example by prefixing a typically small number of subgradient iterations to be performed. To highlight the performance of our approach in view

Table 3 Err%. Small-sized instances

Type	Characteristics			Bundle			IGAP			
	m	n	z_{IP}	$\bar{z}_1$	Err%	Time	$\bar{z}_1$	N_{heu}	Err%	Time
gap1-1*	5	15	336	366.00	8.93%	0.03	339.16	62	0.94%	0.16
gap1-2	5	15	327	366.00	11.93%	0.02	334.05	33	2.16%	0.08
gap1-3*	5	15	339	367.00	8.26%	0.02	343.80	41	1.42%	0.13
gap1-4	5	15	341	368.00	7.92%	0.02	346.16	44	1.51%	0.06
gap1-5	5	15	326	365.00	11.96%	0.02	330.36	46	1.34%	0.03
gap2-1	5	20	434	484.00	11.52%	0.05	439.05	73	1.16%	0.39
gap2-2	5	20	436	485.00	11.24%	0.02	443.73	53	1.77%	0.06
gap2-3	5	20	420	484.00	15.24%	0.02	424.82	71	1.15%	0.34
gap2-4	5	20	419	485.00	15.75%	0.02	426.46	49	1.78%	0.06
gap2-5	5	20	428	482.00	12.62%	0.06	429.40	74	0.33%	0.42
gap3-1	5	25	580	612.00	5.52%	0.03	582.28	100	0.39%	1.25
gap3-2	5	25	564	607.00	7.62%	0.02	568.07	86	0.72%	0.78
gap3-3	5	25	573	607.00	5.93%	0.05	576.13	92	0.55%	1.17
gap3-4	5	25	570	609.00	6.84%	0.02	572.95	103	0.52%	1.39
gap3-5	5	25	564	608.00	7.80%	0.03	567.87	88	0.69%	0.61
gap4-1	5	30	656	724.00	10.37%	0.02	662.31	97	0.96%	1.28
gap4-2*	5	30	644	728.00	13.04%	0.02	655.80	82	1.83%	0.61
gap4-3	5	30	673	724.00	7.58%	0.05	679.51	87	0.97%	0.72
gap4-4	5	30	647	723.00	11.75%	0.05	652.20	108	0.80%	1.75
gap4-5	5	30	664	725.00	9.19%	0.03	667.57	97	0.54%	1.11
gap5-1	8	24	563	588.00	4.44%	0.02	567.27	112	0.76%	2.03
gap5-2	8	24	558	588.00	5.38%	0.06	562.63	110	0.83%	1.84
gap5-3	8	24	564	588.00	4.26%	0.02	567.08	112	0.55%	1.89
gap5-4	8	24	568	589.00	3.70%	0.05	573.22	107	0.92%	1.55
gap5-5	8	24	559	587.00	5.01%	0.02	565.52	87	1.17%	0.80
gap6-1	8	32	761	786.00	3.29%	0.02	765.41	130	0.58%	3.44
gap6-2*	8	32	759	788.00	3.82%	0.02	762.65	133	0.48%	3.30
gap6-3	8	32	758	784.00	3.43%	0.03	762.65	125	0.61%	2.92
gap6-4	8	32	752	788.00	4.79%	0.02	756.53	121	0.60%	2.59
gap6-5	8	32	747	784.00	4.95%	0.03	751.87	139	0.65%	3.63
gap7-1	8	40	942	980.00	4.03%	0.02	945.66	185	0.39%	7.36
gap7-2	8	40	949	985.00	3.79%	0.03	952.94	183	0.42%	6.14
gap7-3	8	40	968	992.00	2.48%	0.03	971.82	178	0.40%	5.63
gap7-4	8	40	945	989.00	4.66%	0.05	948.55	158	0.38%	4.77
gap7-5	8	40	951	981.00	3.15%	0.02	955.69	154	0.49%	4.58
gap8-1*	8	48	1133	1178.00	3.97%	0.05	1137.90	164	0.43%	4.95
gap8-2	8	48	1134	1182.00	4.23%	0.05	1139.18	187	0.46%	6.59

Table 3 (*Continued*)

Type	Characteristics			Bundle			IGAP			
	m	n	z_{IP}	$\bar{z}_1$	$Err\%$	Time	$\bar{z}_1$	N_{heu}	$Err\%$	Time
gap8-3	8	48	1141	1181.00	3.51%	0.02	1144.96	200	0.35%	6.45
gap8-4	8	48	1117	1180.00	5.64%	0.03	1122.94	234	0.53%	8.53
gap8-5*	8	48	1127	1174.00	4.17%	0.02	1133.21	142	0.55%	3.88
gap9-1	10	30	709	737.00	3.95%	0.02	713.84	132	0.68%	3.19
gap9-2	10	30	717	739.00	3.07%	0.03	722.76	112	0.80%	2.11
gap9-3	10	30	712	737.00	3.51%	0.02	718.98	93	0.98%	1.89
gap9-4	10	30	723	739.00	2.21%	0.03	726.46	126	0.48%	2.83
gap9-5	10	30	706	735.00	4.11%	0.05	714.46	87	1.20%	1.91
gap10-1	10	40	958	986.00	2.92%	0.02	960.78	216	0.29%	7.48
gap10-2	10	40	963	986.00	2.39%	0.05	967.20	156	0.44%	4.64
gap10-3	10	40	960	986.00	2.71%	0.05	964.09	181	0.43%	5.91
gap10-4	10	40	947	982.00	3.70%	0.03	950.03	248	0.32%	8.98
gap10-5	10	40	947	983.00	3.80%	0.02	953.20	167	0.65%	4.98
gap11-1*	10	50	1139	1214.00	6.58%	0.05	1142.97	229	0.35%	9.03
gap11-2	10	50	1178	1219.00	3.48%	0.06	1182.76	190	0.40%	6.86
gap11-3*	10	50	1195	1224.00	2.43%	0.03	1196.33	241	0.11%	9.06
gap11-4**	10	50	1171	1219.00	4.10%	0.02	1176.71	100	0.49%	1.58
gap11-5*	10	50	1171	1223.00	4.44%	0.02	1175.34	221	0.37%	8.53
gap12-1	10	60	1451	1479.00	1.93%	0.02	1453.32	295	0.16%	11.55
gap12-2*	10	60	1449	1485.00	2.48%	0.02	1452.22	259	0.22%	11.27
gap12-3	10	60	1433	1480.00	3.28%	0.02	1436.87	278	0.27%	10.80
gap12-4	10	60	1447	1481.00	2.35%	0.03	1449.31	423	0.16%	18.16
gap12-5*	10	60	1446	1485.00	2.70%	0.03	1449.45	260	0.24%	9.83

of possible use in such context we report in Tables 3 and 4 the results obtained by both our algorithm and the standard bundle method for a fixed number of exact Lagrangian relaxation solutions. In particular, we report, for each small-sized instance (see Table 3), the value $\bar{z}_1$ of the bound obtained after one call of a standard bundle algorithm, along with the percentage error $\text{Err}\% = \frac{z_{IP}-\bar{z}_1}{z_{IP}}$, and for each large-sized instance (see Table 4), the value $\bar{z}_{10}$ of the bound obtained after 10 calls of a standard bundle algorithm, along with the percentage error $\text{Err}\% = |\frac{z_{IP}-\bar{z}_{10}}{z_{IP}}|$. For the incremental case $\bar{z}_1$ and $\bar{z}_{10}$ indicate the value of the bound obtained after exactly solving the m knapsack problems once and 10 times, respectively, while N_{heu} is the corresponding number of heuristic calls. The results in Tables 3 and 4 confirm that the use of an incremental approach can be highly effective when embedded in a branch-and-bound context. Indeed, the reduction of the error with respect to a standard bundle algorithm is noticeable, often being one order of magnitude smaller.

Table 4 Err%. Type A–E instances

Type	Characteristics			Bundle			IGAP			
	m	n	z_{IP}	$\bar{z}_1$	Err%	Time	$\bar{z}_1$	N_{heu}	Err%	Time
A	5	100	1698	1524.82	10.20%	0.02	1696.08	81	0.11%	0.02
A	5	200	3235	2763.34	14.58%	0.03	3232.97	104	0.06%	0.03
A	10	100	1360	1095.83	19.42%	0.02	1356.23	104	0.28%	0.05
A	10	200	2623	2192.51	16.41%	0.03	2620.11	120	0.11%	0.05
A	20	100	1158	703.04	39.29%	0.03	1156.88	128	0.10%	0.06
A	20	200	2339	1672.29	28.50%	0.06	2335.42	157	0.15%	0.67
B	5	100	1843	1482.46	19.56%	0.02	1808.09	165	1.89%	0.05
B	5	200	3552	3056.30	13.96%	0.02	3469.55	159	2.32%	0.09
B	10	100	1407	1160.40	17.53%	0.06	1398.64	115	0.59%	0.05
B	10	200	2827	2299.08	18.67%	0.03	2794.55	215	1.15%	0.05
B	20	100	1166	948.53	18.65%	0.05	1159.58	145	0.55%	0.16
B	20	200	2339	1794.08	23.30%	0.06	2330.32	183	0.37%	0.88
C	5	100	1931	1596.03	17.35%	0.03	1900.79	146	1.56%	0.08
C	5	200	3456	2948.70	14.68%	0.05	3368.79	147	2.52%	0.05
C	10	100	1402	1168.70	16.64%	0.02	1382.43	175	1.40%	0.05
C	10	200	2806	2445.68	12.84%	0.05	2772.92	227	1.18%	0.06
C	20	100	1243	945.88	23.90%	0.03	1220.71	147	1.79%	0.16
C	20	200	2391	1956.84	18.16%	0.08	2374.63	282	0.68%	1.55
D	5	100	6353	2612.87	58.87%	0.03	5122.56	109	19.37%	0.02
D	5	200	12742	5288.98	58.49%	0.05	5466.10	72	57.10%	0.06
D	10	100	6348	2465.29	61.16%	0.05	6187.88	189	2.52%	0.06
D	10	200	12432	4521.73	63.63%	0.05	11974.72	271	3.68%	0.19
D	20	100	6190	2253.02	63.60%	0.06	5352.97	79	13.52%	0.09
D	20	200	12241	4237.98	65.38%	0.05	10308.13	157	15.79%	0.80
E	5	100	12681	3758.21	70.36%	0.03	8992.45	131	29.09%	0.08
E	5	200	24930	7357.53	70.49%	0.06	18111.32	202	27.35%	0.03
E	10	100	11577	3502.62	69.74%	0.03	5945.94	78	48.64%	0.09
E	10	200	23307	7076.62	69.64%	0.06	17582.21	228	24.56%	0.09
E	20	100	8436	3216.49	61.87%	0.08	6302.06	137	25.30%	0.28
E	20	200	22379	6878.79	69.26%	0.06	11808.51	132	47.23%	0.53

Acknowledgements We would like to thank Paolo Toth for his insightful comments on the first version of this paper. We are also grateful to two anonymous referees for their careful reading of the manuscript, and for further pushing us toward a deeper numerical experimentation of the algorithm.

References

1. Beasley, J.: OR-library, distributing test problems by electronic mail. J. Oper. Res. Soc. **41**, 1069–1072 (1990)

2. Bertsekas, D., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific, Belmont (1996)
3. Cattrysse, D.G., Van Wassenhove, L.N.: A survey of algorithms for the generalized assignment problem. Eur. J. Oper. Res. **60**, 260–272 (1992)
4. Fisher, M.L.: The Lagrangean relaxation method for solving integer programming problems. Manag. Sci. **27**, 1–18 (1981)
5. Fisher, M.L., Jaikumar, R., Van Wassenhove, L.N.: A multiplier adjustment method for the generalized assignment problem. Manag. Sci. **32**, 1095–1103 (1986)
6. Gaudioso, M., Giallombardo, G., Miglionico, G.: An incremental method for solving convex finite min-max problems. Math. Oper. Res. **31**, 173–187 (2006)
7. Geoffrion, A.M.: Lagrangean relaxation and its uses in integer programming. Math. Program. Study **2**, 82–114 (1974)
8. Goldfarb, D., Idnani, A.: A numerically stable dual method for solving strictly convex quadratic program. Math. Program. **27**, 1–33 (1983)
9. Grippo, L.: Convergent on-line algorithms for supervised learning in neural networks. IEEE Trans. Neural Netw. **11**, 1284–1295 (2000)
10. Hiriart-Urruty, J., Lemaréchal, C.: Convex Analysis and Minimization Algorithms, vols. I, II. Springer, New York (1993)
11. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley, New York (1990)
12. Nauss, R.M.: Solving the generalized assignment problem: an optimizing and heuristic approach. INFORMS J. Comput. **15**, 249–266 (2003)
13. Nedic, A., Bertsekas, D.P.: Incremental subgradient methods for nondifferentiable optimization. SIAM J. Optim. **12**, 109–138 (2001)
14. Nemhauser, G., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
15. Savelsbergh, M.: A branch and price algorithm for the generalized assignment problem. Oper. Res. **45**, 831–841 (1997)
16. Yagiura, M., Ibaraki, T., Glover, F.: An ejection chain approach for the generalized assignment problem. INFORMS J. Comput. **16**, 133–151 (2004)
17. Yagiura, M., Ibaraki, T., Glover, F.: A path relinking approach with ejection chains for the generalized assignment problem. Eur. J. Oper. Res. **169**, 548–569 (2006)