HAGER-ZHANG ACTIVE SET ALGORITHM FOR LARGE-SCALE CONTINUOUS KNAPSACK PROBLEMS

By R.Tavakoli

Sharif University of Technology

The structure of many real-world optimization problems includes minimization of a nonlinear (or quadratic) functional subject to bound and singly linear constraints (in the form of either equality or bilateral inequality) which are commonly called as continuous knapsack problems. Since there are efficient methods to solve large-scale bound constrained nonlinear programs, it is desirable to adapt these methods to solve knapsack problems, while preserving their efficiency and convergence theories. The goal of this paper is to introduce a general framework to extend a box-constrained optimization solver to solve knapsack problems. This framework includes two main ingredients which are O(n) methods; in terms of the computational cost and required memory; for the projection onto the knapsack constrains and the null-space manipulation of the related linear constraint. The main focus of this work is on the extension of Hager-Zhang active set algorithm (SIAM J. Optim. 2006, pp. 526–557). The main reasons for this choice was its promising efficiency in practice as well as its excellent convergence theories (e.g., superlinear local convergence rate without strict complementarity assumption). Moreover, this method does not use any explicit form of second order information and/or solution of linear systems during iteration which makes it an ideal for large-scale problems. The efficient implementation of the method is discussed in details.

1. Introduction. The goal of this paper is to develop efficient methods to solve large-scale linearly constrained optimization problems with the following structure

$$\min f(\mathbf{x}) \quad \texttt{s.t.} \quad \mathbf{x} \in \mathcal{D} \tag{1.1}$$

AMS 2000 subject classifications: 90C06, 90C26, 65Y20

Keywords and phrases: degenerate constraints, gradient projection, linearly constrained optimization, null space method, singly linear bound constrained optimization, superlinear convergence, support vector machines

the feasible set $\mathcal{D}$ is equal to either of $\mathcal{D}_{\mathcal{E}}$ or $\mathcal{D}_{\mathcal{I}}$ which are defined as follows

$$\mathcal{D}_{\mathcal{E}} \overset{def}{=} \{\mathbf{x} \in \mathcal{B} : \quad \mathbf{a}^T\mathbf{x} = b \ \} \tag{1.2}$$

$$\mathcal{D}_{\mathcal{I}} \overset{def}{=} \{\mathbf{x} \in \mathcal{B} : \quad b_l \leqslant \mathbf{a}^T\mathbf{x} \leqslant b_u\} \tag{1.3}$$

$$\mathcal{B} \overset{def}{=} \{\mathbf{x} \in \mathbb{R}^n : \quad \mathbf{l} \leqslant \mathbf{x} \leqslant \mathbf{u}\} \tag{1.4}$$

where f is a real-valued continuously differentiable function defined on $\mathcal{D}$, $\mathbf{a}, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n$, $b, b_l, b_u \in \mathbb{R}$ and $\mathbf{l} \leqslant \mathbf{u}$. In the literature the constraint set $\mathcal{D}$ is usually called as the continuous knapsack constraints. Although problem (1.1) can be studied under context of general linearly constrained optimization problems, due to the importance and wide range of applications, there are a lot of works specifically aligned for the solution of (1.1), e.g. see: [Moré and Vavasis, 1990; Melman and Rabinowitz, 2000; Dai and Fletcher, 2006; Dahiya et al., 2007; Lucidi et al., 2007; Lin et al., 2009; Tseng and Yun, 2008; Gonzalez-Lima et al., 2009]. Optimization problems like (1.1) occur frequently in the filed of operational researchs like optimal resource allocation and marketing, refer to [Bretthauer and Shetty, 2002; Patriksson, 2008] as some topical reviews.

Although in the past two decades, several methods with good convergence theories have been introduced to solve linearly constrained optimization problems, developing an efficient method to solve large scale problems (in terms of computational cost and memory usage) is still remained as an open problem. An O(n) growth of the memory usage and computational cost per iteration as well as the global convergence and quadratic or superlinear local convergence seems to be desirable properties of an efficient method. To realize this goal it is required to avoid solutions of linear systems of equations per iteration or at least avoiding the exact solution of such systems. This goal is almost realized in the case of bound-constrained optimization problems and currently lot of methods are available to solve large scale box-constrained optimizations. In this context we can recall the active set Newton algorithm of Facchinei et al. [2002], affine-scaling interior-point Newton methods of Heinkenschloss et al. [1999] and new active set algorithm of Hager and Zhang [2006]. One of the most important properties of these methods is relaxing the strict complementarity condition to the strong second order sufficient optimality condition to prove the local superlinear rate of convergence.

It seems that, a key to develop an efficient method to solve large scale knapsack constrained optimization problems is to adapt large scale box-constrained solvers for such problems; while preserving their efficiency and convergence theories. There were a few works in which this clue was taken into account. In [Dai and Fletcher, 2006], by introducing an efficient method to project a trial step on to the knapsack constraints, the projected Barzilai-Borwein method [Dai and Fletcher, 2005] was extended to solve knapsack constrained optimization problems. Modifying the search direction to keep iterations interior with respect to the related linear constraint, Gonzalez-Lima et al. [2009] extended the affine-scaling interior-point CBB method [Hager et al., 2009] to solve knapsack constrained optimization problems. However, the mentioned methods possess a linear rate of local convergence at the best condition.

Roughly speaking, the goal of this paper is to introduce a general framework to extend almost every bound-constrained optimization method to solve knapsack constrained problems without destroying its efficiency and convergence theories. For this purpose we shall specifically concentrate on new active set algorithm by Hager and Zhang [2006]. The main reasons for this choice are its excellent convergence and efficiency in contrast to alternative methods. Besides, following the presented method, hopefully, it may be possible to extend other methods without additional technical difficulties.

The efficient and stable implementation of the projection onto the knapsack constrains and null-space treatment of the linear constraint are main ingredients of the presented framework for mentioned extension in this study. Our method to project a point onto the space of knapsack constrains is partly identical to that of [Dai and Fletcher, 2006; Kiwiel, 2008]. But, the technical details are different and some new results are also included. According to our knowledge, an efficient and stable implementation of the null-space method for knapsack constrains is introduced in this study for the first time.

Notations. For any scalar $v \in \mathbb{R}$, $v^+ = \max\{v, 0\}$ and $v^- = \min\{v, 0\}$. The median operator acting on triple $\{u, v, w\}$ is denoted by $\texttt{mid}(u, v, w)$, i.e., $\texttt{mid}(u, v, w) = \max\{u, \min\{v, w\}\}$. For an arbitrary vector $\mathbf{v} \in \mathbb{R}^n$ and bound vectors $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ ($\mathbf{l} \leqslant \mathbf{u}$), the operator $\texttt{mid}(\mathbf{l}, \mathbf{v}, \mathbf{u})$ results a vector $\mathbf{w} \in \mathbb{R}^n$ such that $w_i = \texttt{mid}(l_i, v_i, u_i),\ i = 1, \dots, n$. The gradient and Hessian of the objective function $f(\mathbf{x})$ with respect to $\mathbf{x}$ are denoted by $\nabla_{\mathbf{x}} f(\mathbf{x})$ and $\nabla^2_{\mathbf{x}} f(\mathbf{x})$ respectively. For any $\mathbf{x} \in \mathcal{B}$ the active and inactive index-set (with respect to bound constraints) are denoted by $\mathcal{A}(\mathbf{x})$ and $\mathcal{I}(\mathbf{x})$ respectively, i.e.

$$\mathcal{A}(\mathbf{x}) = \{i \in [0, n] :\ x_i = l_i \texttt{ or } x_i = u_i\}$$
$$\mathcal{I}(\mathbf{x}) = \{i \in [0, n] :\ l_i < x_i < u_i\}$$

The operator $\texttt{dim}\ (\mathcal{A}(\mathbf{x}))$ returns the dimension of the active set with respect to bound constraints at $\mathbf{x}$. Assuming $\texttt{dim}\ (\mathcal{A}(\mathbf{x})) = t$, the operator $\texttt{shrink}(\mathbf{x})$

gives the row (column) vector $\mathbf{x} \in \mathbb{R}^n$ as the input argument and returns the reduced row (column) vector $\mathbf{v} \in \mathbb{R}^{n-t}$ which includes only entries of $\mathbf{x}$ corresponding to the inactive indices. Similarly, the operator $\texttt{expand}(\mathbf{v})$ gives the row (column) vector $\mathbf{v} \in \mathbb{R}^{n-t}$ as the input argument and returns the expanded row (column) vector $\mathbf{x} \in \mathbb{R}^n$ by adding the active indices to $\mathbf{v}$ accordingly. The subscript k ($\square_k$) is often used to denote the quantity of interest at the k-th iteration. When there is no confusion, we may use $\mathbf{v}_k$ to denote $\mathbf{v}(\mathbf{x}_k)$.

2. Projection onto the knapsack constraints.

As it was mentioned in section 1, the projection of a trial point onto the feasible set ($\mathcal{D}$) is one of the main ingredient of the presented method in this study. This issue is studied in this section. Consider the trial point $\mathbf{y} \in \mathbb{R}^n$, the projection of $\mathbf{y}$ onto the feasible set; which is denoted by $\mathbf{z} \in \mathbb{R}^n$ here; is equal to solution of the following constrained least square problem

$$\mathbf{z} := \mathcal{P}_{\mathcal{D}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{D}} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 \tag{2.1}$$

$\mathcal{P}_{\mathcal{D}}(\mathbf{y})$ in (2.1) denotes the projection operator which projects a trial point $\mathbf{y}$ onto the feasible set $\mathcal{D}$ with respect to the euclidean norm. In the first part of this section we shall consider projection onto $\mathcal{D}_{\mathcal{E}}$. Later we extend our method to solve problem of projection onto $\mathcal{D}_{\mathcal{I}}$. Solution of (2.1) with $\mathcal{D} = \mathcal{D}_{\mathcal{E}}$ is equivalent to solving the following quadratic programming (QP) problem

$$\min \quad \frac{1}{2}\,\mathbf{x}^T\mathbf{I}\,\mathbf{x} - \mathbf{y}^T\mathbf{x} \quad \texttt{s.t.}: \quad \mathbf{a}^T\mathbf{x} = b, \quad \mathbf{l} \leqslant \mathbf{x} \leqslant \mathbf{u}, \tag{2.2}$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ denotes the identity matrix. In general solution of a QP problem is expensive, however, exploiting specific structure of (2.2) results an efficient computational method which is discussed here.

PROPOSITION 2.1. *The feasible set $\mathcal{D}_{\mathcal{E}}$ is non-empty if the following conditions hold*

$$\sum_{i=1}^{n}(u_i a_i^- + l_i a_i^+) \leqslant b \leqslant \sum_{i=1}^{n}(u_i a_i^+ + l_i a_i^-)$$

PROOF. The proof is trivial considering the geometry of $\mathcal{D}_{\mathcal{E}}$ (also see equation 2.6 in [Dai and Fletcher, 2006]). $\square$

THEOREM 2.2. *Assume that the feasible domain $\mathcal{D}_{\mathcal{E}}$ is non-empty. Then problem* (2.2) *has a unique solution* $\mathbf{x}^*$ *which is computed by*

$$\mathbf{x}^* = \texttt{mid}(\mathbf{l},\ \mathbf{y} - \lambda^* \mathbf{a},\ \mathbf{u}) \tag{2.3}$$

where $\lambda^* \in \mathbb{R}^n$ *is the Lagrange multiplier corresponding to linear equality constraint* $\mathbf{a}^T\mathbf{x} = b$ *at the unique KKT point of* (2.2), *and* λ^* *is equal to the unique root of the following equation*

$$h(\lambda) = b - \sum_{i=1}^{n} \left[\ a_i \ \texttt{mid}(l_i,\ y_i - \lambda a_i,\ u_i)\ \right].$$

PROOF. The existence and uniqueness of solution and Lagrange multiplier is followed directly by the strict convexity of the problem and non-emptiness assumption of the feasible domain. Now let us to reformulate (2.2) as a box-constrained optimization problem by augmenting the linear equality constraint to the objective function

$$\mathcal{L}[\mathbf{x};\lambda] = \frac{1}{2}\ \mathbf{x}^T\mathbf{I}\ \mathbf{x} - \mathbf{y}^T\mathbf{x} + \lambda(\mathbf{a}^T\mathbf{x} - b) \quad \texttt{s.t.:} \quad \mathbf{l} \leqslant \mathbf{x} \leqslant \mathbf{u} \tag{2.4}$$

where $\lambda \in \mathbb{R}$ is the lagrange multiplier corresponding to the linear equality constraint. It is obvious that the unique stationary point of Lagrangian $\mathcal{L}[\mathbf{x};\lambda]$ constrained by box $\mathcal{B}$ which is shown by $(\mathbf{x}^*, \lambda^*)$ is equal to the solution of (2.2).

For a fixed value of λ, the box constrained stationary point of (2.4), $\mathbf{x}^*(\lambda)$, can be computed by the projected gradient method (cf. [Lin and Moré, 1999]). Using optimality conditions based on the projected gradient method, $\mathbf{x}^*(\lambda)$ is equal to the unique zero of projected gradient (with respect to $\mathbf{x}$) of Lagrangian $\mathcal{L}$, i.e.,

$$\mathcal{P}_{\mathcal{B}}\big(\mathbf{x}^*(\lambda) - \nabla_{\mathbf{x}}\mathcal{L}[\mathbf{x}^*(\lambda);\lambda]\big) - \mathbf{x}^*(\lambda) = 0 \tag{2.5}$$

where $\mathcal{P}_{\mathcal{B}}(\cdot)$ denotes the projection operator onto $\mathcal{B}$. Simplification of (2.5) results

$$\mathcal{P}_{\mathcal{B}}\big(\mathbf{y} - \lambda\mathbf{a}\big) - \mathbf{x}^*(\lambda) = 0 \tag{2.6}$$

since for an arbitrary vector $\mathbf{v} \in \mathbb{R}^n$, $\mathcal{P}_{\mathcal{B}}(\mathbf{v})$ is equal to $\texttt{mid}(\mathbf{l}, \mathbf{v}, \mathbf{u})$, the solution of equation (2.6) can be written in the following explicit form

$$\mathbf{x}^*(\lambda) = \texttt{mid}(\mathbf{l},\ \mathbf{y} - \lambda\mathbf{a},\ \mathbf{u}) \tag{2.7}$$

considering the necessary optimality conditions of (2.2), λ^* is equal to unique zero of the following non-smooth equation

$$h(\lambda) = b - \sum_{i=1}^{n} a_i x_i^*(\lambda) \tag{2.8}$$

which completes the proof. $\square$

Therefore, finding the unique solution of single parameter non-smooth equation $h(\lambda) = 0$ is the main step to solve (2.2). Function $h(\lambda)$ is a piecewise linear function with at most $2n$ breakpoints. The corresponding λ of these breakpoints can be shown by set

$$\mathcal{T}_\lambda = \{\ \lambda_i^l,\ \lambda_i^u \mid i = 1, \ldots, n;\ a_i \neq 0\}$$

where

$$\lambda_i^l = (y_i - l_i)/a_i, \quad \lambda_i^u = (y_i - u_i)/a_i, \quad i = 1, \ldots, n,$$

it is evident that $\lambda_i^u \leqslant \lambda_i^l$ for $a_i > 0$ and $\lambda_i^l \leqslant \lambda_i^u$ for $a_i < 0$. Considering λ coordinates of breakpoints, for $a_i > 0$, $x_i(\lambda)$ can be expressed in the following form,

$$x_i(\lambda) = \begin{cases} u_i, & \texttt{if} \quad \lambda \leqslant \lambda_i^u, \\ y_i - \lambda a_i, & \texttt{if} \quad \lambda_i^u \leqslant \lambda \leqslant \lambda_i^l, \\ l_i, & \texttt{if} \quad \lambda \geqslant \lambda_i^l. \end{cases} \tag{2.9}$$

in the same way for $a_i < 0$, we have

$$x_i(\lambda) = \begin{cases} u_i, & \texttt{if} \quad \lambda \geqslant \lambda_i^u, \\ y_i - \lambda a_i, & \texttt{if} \quad \lambda_i^l \leqslant \lambda \leqslant \lambda_i^u, \\ l_i, & \texttt{if} \quad \lambda \leqslant \lambda_i^l. \end{cases} \tag{2.10}$$

note that for $a_i = 0$, x_i is independent from λ and is equal to $\texttt{mid}(l_i,\ y_i,\ u_i)$. In this case, it is possible to consider a reduced counterpart of the original problem. Therefore, without loss of generality, we can consider $a_i \neq 0$. Considering (2.9) and (2.9), $x_i(\lambda)$ is a continuous piecewise linear and monotonically nonincreasing function of λ for $a_i > 0$; and in the same way; $x_i(\lambda)$ is a continuous piecewise linear and monotonically nondecreasing function of λ for $a_i < 0$. Therefore, according to (2.7) and (2.8), $h(\lambda)$ is a continuous piecewise linear and monotonically nonincreasing function of λ. It is obvious that finding the root of (2.8) is the most expensive part of projection in this section.

In [Kiwiel, 2008] a median based breakpoint searching algorithm was introduced to find the unique root of (2.8). This algorithm is somehow bisecting the set of breakpoints (consider the traditional interval bisection method). Although, by using O(n) median finding method, the computational complexity of this algorithm is O(n), the proportionality constant and the amount of operations per step is relatively large. In [Dai and Fletcher, 2006] a two-phase algorithm was suggested to find the root of (2.8). In the first stage, the bracketing phase, an interval $[\lambda_L, \lambda_R]$ where $h(\lambda_L) \cdot h(\lambda_R) < 0$ is found. Then in the second phase, secant step, the root of (2.8) is computed by an accelerated secant algorithm. In the present study, we use a combination of the bisection and quadratic interpolation methods to find the root of (2.8).

Proposition 2.3. *Assume* $\lambda_L = \min\{\ \lambda_i^l,\ \lambda_i^u \mid i = 1, \ldots, n;\ a_i \neq 0\}$ *and* $\lambda_R = \max\{\ \lambda_i^l,\ \lambda_i^u \mid i = 1, \ldots, n;\ a_i \neq 0\}$ *then* $h(\lambda_L) \cdot h(\lambda_R) < 0$ *or* $h(\lambda_L) \cdot h(\lambda_R) = 0$*, i.e., the root of* (2.8) *happens within the interval* (λ_L, λ_R) *or it is equal to either of* λ_L *and* λ_R*. Moreover,* $h(\lambda) \leqslant 0$ *for* $\lambda \leqslant \lambda_L$ *and* $h(\lambda) \geqslant 0$ *for* $\lambda \geqslant \lambda_R$*.*

Proof. Since $h(\lambda)$ is a continuous piecewise linear and monotonically nonincreasing function of λ and it has a unique root, we should have $h(\lambda) \leqslant 0$ for $\lambda \leqslant \lambda_L$ and $h(\lambda) \geqslant 0$ for $\lambda \geqslant \lambda_R$. So the root of $h(\lambda)$ happens within the interval $[\lambda_L, \lambda_R]$. □

Considering the Proposition 2.3, the bracketing phase used in [Dai and Fletcher, 2006] is not required as it is a-priori known. Therefore, it is possible to find the root of $h(\lambda)$ by the traditional interval bisection method.

Assume that the error at the k-th step of the bisection algorithm is shown by ϵ_k, where $\epsilon_0 = (\lambda_R - \lambda_L)/2$. Then $\epsilon_{k+1} = \epsilon_k/2$. Therefore the number of bisection steps to find λ^* within tolerance ϵ is equal to $\log_2(\epsilon_0/\epsilon)$. Considering the limited arithmetic precision of computers, the upper bound on number of bisection steps is priori known and is independent from n. Notice that the number of set bisection in [Kiwiel, 2008] is function of n. Therefore, for large n, the worst case complexity of the interval bisection method should be better than the worst case complexity of the set bisection algorithm of [Kiwiel, 2008]. Moreover, when the interval became sufficiently small (in the interval bisection algorithm), it is possible to do a linear interpolation between three consecutive available points to find a trial root. Then discarding the algorithm if the trial root is zero of $h(\lambda)$.

In contrast to alternative methods, the bisection method is the most robust algorithm and is enable to find the root within the machine precision. However, its linear convergence is cumbersome. Similar to [Brent, 2002], a combination of the bisection and the inverse quadratic interpolation methods is used in the present study to improve the convergence-rate, simultaneously preserving the robustness of the algorithm. Prior to introducing this method, it is worth to state the following remark to decrease the computational complexity (hopefully with a $\log_2$ slope) in the course of iterations. Assume that the current search interval is shown by $[\lambda_l, \lambda_r]$.

Remark 2.4. Considering (2.9) and (2.10) together with the current value of $[\lambda_l, \lambda_r]$, it is possible to reduce the computational cost by freezing the components of $\mathbf{x}$ vector for which x_i is determined. More clearly, when $a_i > 0$, $x_i^*(\lambda)$ can be fixed to u_i or l_i if $\lambda_r \leqslant \lambda_i^u$ or $\lambda_l \geqslant \lambda_i^l$ respectively; and when $a_i < 0$, $x_i^*(\lambda)$ can be fixed to u_i or l_i if $\lambda_l \leqslant \lambda_i^u$ or $\lambda_r \geqslant \lambda_i^l$ respectively.

Consider $\epsilon_M \in \mathbb{R}^+$ as the machine precision, we are looking for the unique root of $h(\lambda)$ within precision $\epsilon \in \mathbb{R}^+$ ($\epsilon > \epsilon_M$). The machine precision can be computed by algorithm 665 of ACM TOMS [Cody, 1988]. In the root finding algorithm in this study we have three points λ_a, λ_b and λ_c such that $h(\lambda_b) \cdot h(\lambda_c) \leqslant 0$, $|h(\lambda_b)| \leqslant |h(\lambda_c)|$, and λ_a may coincide with λ_c. Initially $\lambda_b = \lambda_L$, $\lambda_c = \lambda_R$ and $\lambda_a = \lambda_c$. The point λ_b is considered as the best approximation to λ^* in the course of iterations.

Consider $\Delta_\lambda = (\lambda_c - \lambda_b)/2$. If $\Delta_\lambda \leqslant \epsilon$ the value of λ_b is returned as an approximation to λ_*, else an inverse quadratic interpolation is used to compute a trial approximation for the root of function h. For the convenience, $h(\lambda_x)$ is shown by h_x henceforth. Assume we have three distinct points (λ_a, h_a), (λ_b, h_b) and (λ_c, h_c), then the quadratic lagrange interpolation formulae can be written as follows

$$\lambda = \frac{(h_\lambda - h_b)(h_\lambda - h_c)}{(h_a - h_b)(h_a - h_c)}\,\lambda_a + \frac{(h_\lambda - h_c)(h_\lambda - h_a)}{(h_b - h_c)(h_b - h_a)}\,\lambda_b + \frac{(h_\lambda - h_a)(h_\lambda - h_b)}{(h_c - h_a)(h_c - h_b)}\,\lambda_c$$

The root of this quadratic approximation, λ_t, can be written in the following explicit form

$$\lambda_t = \lambda_b + p/q \tag{2.11}$$

where $r = h_b/h_c$, $s = h_b/h_a$, $t = h_a/h_c$, $q = \pm(t-1)(r-1)(s-1)$ and $p = \pm st(r-t)(\lambda_c - \lambda_b) - s(1-r)(\lambda_b - \lambda_a)$. At the start of iterations in which there are only two distinct points, a linear interpolation is used to find the trial root. When $q \approx 0$ the overflow problem may cause to the failure of computation. Therefore, the trial root λ_t will be rejected in this case. Moreover, when the interpolating parabola has two roots between λ_b and λ_c or when its root is located outside of this interval, the interpolation is poor and the trial root λ_t should be rejected. In the case of inefficient step ($|p| \leqslant \epsilon|q|$) or when the current p/q is greater than half of its previous value, the trial root λ_t will be rejected to avoid the slow convergence. Anyway the trail root which is resulted from the inverse interpolation be rejected, a bisection step will be performed. In summary, if either of the following conditions is met the trial root of the inverse quadratic interpolation is rejected and a bisection step is used instead

$$|p| \geqslant 2/3\ |q\Delta_\lambda|, \quad |p| \leqslant \epsilon|q|, \quad |p/q| \geqslant 1/2\ |p/q|_{old}$$

The stopping criteria is a critical issue to avoid excess computations when rounding errors prevent further progress toward the exact solution in the vicinity of root. Following [Wilkinson, 1994], the following relation is used as the stopping tolerance in this study

$$tol = 2\epsilon_M|\lambda_b| + \epsilon/2$$

Note that the above criterion does not tell us how close we are to the root, but only that we are in some interval about the zero where roundoff error may be dominating our calculations.

According to our numerical experiments, this method finds the desired root within machine precision by a fewer function evaluations in contrast to mentioned alternatives (in our experience the number of function evaluation calls were usually below 12 for double-precision arithmetic and $\epsilon = 1.e-15$).

Remark 2.5. Similar to new line-search algorithm introduced in [Hager and Zhang, 2005], the mentioned root finding method robustly tolerates the limited machine precision; simultaneously remains efficient as much as possible. Therefore, it has a good potential to be adapted as an alternative robust line-search method. In particular, in contrast to that of [Hager and Zhang, 2005] which uses inverse linear interpolations (secant steps), it uses inverse quadratic interpolations without additional function calls.

The following piece of code shows the C implementation of the mentioned root finding method. In this code `h`, `epsm`, `la`, `lb`, `lc`, `ha`, `hb`, `hc`, `del` and `delo` are equivalent to function h, ϵ_M , λ_a, λ_b, λ_c, h_a, h_b, h_c, Δ_λ and the old value of Δ_λ respectively.

```
double find_root (double la, double lb, const double eps) {
  int iter_max = 100, iter = 0; double epsm = 5.e-16;
  double lc = lb, ha = h(la), hb = h(lb), hc = hb;
  double tol, dl, del, delo, p, q, r, s, t;
  for (; iter<iter_max; iter++) {
    if (hb*hc > 0.0) {
      lc = la;  hc = ha; delo = del = lb - la;
    }
    if (fabs (hc) < fabs (hb)) {
      la = lb;  lb = lc;  lc = la;
      ha = hb;  hb = hc;  hc = ha;
    }
    tol = 2.0*epsm*fabs(lb) + 0.5*eps;
    dl = 0.5*(lc-lb);
    if (fabs(dl) <= tol || fabs(hb)<= tol) return lb;
    if (fabs(delo) >= tol && fabs(ha) > fabs(hb)) {
      if (la == lc) {
        s = hb/ha; p = 2.0*dl*s; q = 1.0-s;
      } else {
        r = hb/hc; s = hb/ha; t = ha/hc;
        p = s*(2.0*dl*t*(t-r)-(lb-la)*(r-1.0));
        q = (r-1.0)*(s-1.0)*(t-1.0);
      }
      if (p > 0.0) q = -q; p = fabs(p);
      if ((2.*p<(3.*dl*q-fabs(tol*q)))||(2.*p<fabs(delo*q))){
        delo = del; del = p/q;
      } else delo = del = dl;
    } else delo = del = dl;
    la = lb; ha = hb;
    if (fabs(del) > tol) lb += del;
    else lb += (dl > 0.0 ? tol : -tol);
    hb = h(lb);
  }
  return lb;
}
```

Now, consider the solution of (2.1) with $\mathcal{D} = \mathcal{D}_\mathcal{I}$ which is equivalent to the following QP problem

$$\min \quad \frac{1}{2}\,\mathbf{x}^T\mathbf{I}\,\mathbf{x} - \mathbf{y}^T\mathbf{x} \quad \text{s.t.}: \quad b_l \leqslant \mathbf{a}^T\mathbf{x} \leqslant b_u, \quad \mathbf{l} \leqslant \mathbf{x} \leqslant \mathbf{u}, \tag{2.12}$$

PROPOSITION 2.6. *The feasible set $\mathcal{D}_\mathcal{I}$ is non-empty if the following conditions hold*

$$\sum_{i=1}^{n}(u_i a_i^- + l_i a_i^+) \leqslant b_l \leqslant b_u \leqslant \sum_{i=1}^{n}(u_i a_i^+ + l_i a_i^-)$$

PROOF. The proof is directly followed from proposition 2.6. □

The following theorem provides an elegant method to compute the solution of (2.12) using results of theorem 2.2.

THEOREM 2.7. *Assume that the feasible set $\mathcal{D}_{\mathcal{I}}$ is non-empty. Then problem (2.12) has a unique solution $\mathbf{x}^*$ which is computed by the following relation*

$$\mathbf{x}^* = \mathtt{mid}(\mathbf{x}_L^*,\ \mathbf{y},\ \mathbf{x}_U^*) \tag{2.13}$$

where $\mathbf{x}_L^*, \mathbf{x}_U^* \in \mathbb{R}^n$ are unique solutions of the following QP problems

$$\mathbf{x}_L := \min \quad \frac{1}{2}\,\mathbf{x}^T\mathbf{I}\ \mathbf{x} - \mathbf{y}^T\mathbf{x} \quad \mathtt{s.t.}: \quad \mathbf{a}^T\mathbf{x} = b_l, \quad \mathbf{l} \leqslant \mathbf{x} \leqslant \mathbf{u}, \tag{2.14}$$

$$\mathbf{x}_U := \min \quad \frac{1}{2}\,\mathbf{x}^T\mathbf{I}\ \mathbf{x} - \mathbf{y}^T\mathbf{x} \quad \mathtt{s.t.}: \quad \mathbf{a}^T\mathbf{x} = b_u, \quad \mathbf{l} \leqslant \mathbf{x} \leqslant \mathbf{u}, \tag{2.15}$$

PROOF. The existence and uniqueness of solutions for problems (2.12), (2.14) and (2.15) are directly followed by the strict convexity of related problems and the non-emptiness assumption of $\mathcal{D}_{\mathcal{I}}$.

Same as previous, we reformulate (2.12) as a bound constrained optimization problem by augmenting the linear inequality constraints to the objective function

$$\mathcal{L}[\mathbf{x};\mu_l;\mu_u] = \frac{1}{2}\,\mathbf{x}^T\mathbf{I}\ \mathbf{x} - \mathbf{y}^T\mathbf{x} + \mu_l(\mathbf{a}^T\mathbf{x} - b_l) + \mu_u(\mathbf{a}^T\mathbf{x} - b_u) \quad \mathtt{s.t.}: \quad \mathbf{l} \leqslant \mathbf{x} \leqslant \mathbf{u}$$

where $\mu_l, \mu_u \in \mathbb{R}$ are the lagrange multipliers corresponding to the inequality constraints $\mathbf{a}^T\mathbf{x} \geqslant b_l$ and $\mathbf{a}^T\mathbf{x} \leqslant b_u$ respectively. It is evident that the unique stationary point of Lagrangian $\mathcal{L}[\mathbf{x};\mu_l;\mu_u]$ constrained by box $\mathcal{B}$ which is shown by $(\mathbf{x}^*, \mu_l^*, \mu_u^*)$ is equal to the unique solution of (2.12).

For fixed values of μ_l and μ_u, the bound constrained stationary point of Lagrangian $\mathcal{L}$, i.e., $\mathbf{x}^*(\mu_l, \mu_u)$, can be computed by the projected gradient method. Using optimality conditions based on the projected gradient method, $\mathbf{x}^*(\mu_l, \mu_u)$ is equal to the unique zero of the projected gradient (with respect to $\mathbf{x}$) of Lagrangian $\mathcal{L}$. Using the same way like (2.5) results

$$\mathbf{x}^*(\mu_l, \mu_u) = \mathtt{mid}\big(\mathbf{l},\ \mathbf{y} - (\mu_l + \mu_u)\mathbf{a}, \mathbf{u}\big) \tag{2.16}$$

assuming $\mu = \mu_l + \mu_u$ simplifies (2.16) to the following form

$$\mathbf{x}^*(\mu) = \mathtt{mid}(\mathbf{l},\ \mathbf{y} - \mu\mathbf{a}, \mathbf{u}) \tag{2.17}$$

note that at least either of μ_l or μ_u is zero at the optimal solution, so, as (2.17) shows, knowing $\mu_l+\mu_u$ at the optimal solution is sufficient to uniquely determine μ_l^* and μ_u^*. Considering the necessary optimality conditions of (2.12), the remaining job is to compute the optimal value of μ (which is shown by μ^*) such that the following inequalities are satisfied

$$b_l \leqslant \mathbf{a}^T \mathbf{x}^*(\mu) \leqslant b_u \tag{2.18}$$

or

$$b_l \leqslant \mathbf{a}^T \mathtt{mid}(\mathbf{l},\ \mathbf{y} - \mu\mathbf{a}, \mathbf{u}) \leqslant b_u \tag{2.19}$$

due to the uniqueness of lagrange multiplier μ^* at the optimal solution, the inequality equation (2.19) has only one solution. Without confusion with the previous definition for $h(\lambda)$, assume the following new definition for function $h(\mu)$

$$h(\mu) = \mathbf{a}^T \mathbf{x}^*(\mu)$$

Similar to the previous arguments on function $h(\lambda)$, it is easy to show that $h(\mu)$ is a continuous piecewise linear and monotonically nonincreasing function of μ with at most $2n$ breakpoints (to save space we avoid repetition of similar discussions for $h(\mu)$).

Assume λ_l^* and λ_u^* are optimal lagrange multipliers corresponding to linear equality constraint in problems (2.14) and (2.15) respectively. Considering the mentioned monotonicity of functions $h(\lambda)$ and $h(\mu)$, the following inequalities are identical

$$\lambda_u^* \leqslant \mu^* \leqslant \lambda_l^* \tag{2.20}$$

therefore an appropriate search interval for μ^* is the bracket $[\lambda_u^*, \lambda_l^*]$. Considering (2.3), (2.19) and (2.20), the following inequalities hold

$$\mathbf{x}_L^* \leqslant \mathbf{x}^* \leqslant \mathbf{x}_U^* \tag{2.21}$$

notice that the inequalities are understood componentwise here. Refereing again to the mentioned monotonicity of functions $h(\lambda)$ and $h(\mu)$, when $\mathbf{x}_L^* \leqslant \mathbf{x}^*$ the inequality $\mathbf{a}^T\mathbf{x}^* \geqslant b_l$ always holds. In the same way, the inequality $\mathbf{a}^T\mathbf{x}^* \leqslant b_u$ will be remained always satisfied when $\mathbf{x}^* \leqslant \mathbf{x}_U^*$. Therefore, replacing bound constraints $\mathbf{l}$ and $\mathbf{u}$ respectively with $\mathbf{x}_L^*$ and $\mathbf{x}_U^*$ reduces problem (2.12) to the following bound constrained QP problem

$$\min \quad \frac{1}{2}\,\mathbf{x}^T\mathbf{I}\,\mathbf{x} - \mathbf{y}^T\mathbf{x} \quad \mathtt{s.t.:} \quad \mathbf{x}_L^* \leqslant \mathbf{x} \leqslant \mathbf{x}_U^* \tag{2.22}$$

with the explicit solution

$$\mathbf{x}^* = \texttt{mid}(\mathbf{x}_L^*,\ \mathbf{y},\ \mathbf{x}_U^*)$$

which completes the proof. □

Therefore, it is possible to solve (2.12) in expense of solving two problems with structures similar to (2.2). However, the asymptotic computational cost is not essentially duplicated in this way. This is because of some shared calculations like computing λ_L, λ_R, $h(\lambda_L)$ and $h(\lambda_R)$. Moreover, the trajectory points produced during the solution of (2.15) can be easily used to reduce the initial search bracket to solve (2.14). For instance, if pair (λ_x, h_x) was produced during the solution of (2.15), the corresponding h to λ_x when solving (2.14) is equal to $h_x + b_l - b_u$.

3. Null-space management of linear constraints.

In this section we present an O(n) method (in terms of computational cost and consumed memory) to treat the linear constraint in knapsack problem. For the convenience, the basic idea of the null-space methods is recalled at the first part of this section (for more details see chapter 5 of [Gill et al., 1981]).

Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ be a full rank matrix. The null-space of $\mathbf{A}$ is denoted by

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{p} \in \mathbb{R}^n :\ \mathbf{A}\mathbf{p} = 0\}$$

which is the set of vectors orthogonal to the rows of $\mathbf{A}$. The null-space of $\mathbf{A}$ is a subspace of $\mathbb{R}^n$ with dimension $n-m$ (consider the full-rank assumption of $\mathbf{A}$). Therefore, any linear combination of two vectors in $\mathcal{N}(\mathbf{A})$ is also in $\mathcal{N}(\mathbf{A})$. Any matrix $\mathbf{Z} \in \mathbb{R}^{n\times(n-m)}$ whose columns form a basis for $\mathcal{N}(\mathbf{A})$ can be considered as a null-space matrix for $\mathbf{A}$. It is easy to show that $\mathbf{Z}$ satisfies $\mathbf{A}^T\mathbf{Z} = 0$.

The range-space of $\mathbf{A}$ is defined as a subspace of $\mathbb{R}^n$ with dimension m which is spanned by the columns of the $\mathbf{A}$ (the set of all linear combinations of $\mathbf{A}$ columns). In particular, we are interested in the range space of $\mathbf{A}^T$, defined by

$$\mathcal{R}(\mathbf{A}^T) = \{\mathbf{q} \in \mathbb{R}^n :\ \mathbf{q} = \mathbf{A}\mathbf{r} \quad \texttt{for some} \quad \mathbf{r} \in \mathbb{R}^m\}$$

It is easy to show that $\mathcal{N}(\mathbf{A})$ and $\mathcal{R}(\mathbf{A}^T)$ are orthogonal subspaces. Therefore, it is possible to uniquely decompose an arbitrary vector $\mathbf{x} \in \mathbb{R}^n$ into sum of the range-space, $\mathbf{q} \in \mathcal{R}(\mathbf{A}^T)$, and null-space, $\mathbf{p} \in \mathcal{N}(\mathbf{A})$, components

$$\mathbf{x} = \mathbf{q} + \mathbf{p} = \mathbf{q} + \mathbf{Z}\mathbf{v}$$

where $\mathbf{v} \in \mathbb{R}^{(n-m)}$. Consider linear system of equations $\mathbf{A}^T\mathbf{x} = \mathbf{b}$ ($\mathbf{b} \in \mathbb{R}^m$). Assuming $\mathbf{x}_0 \in \mathbb{R}^n$ as a particular solution for $\mathbf{A}^T\mathbf{x} = \mathbf{b}$, any other solution $\mathbf{x}$ can be parameterized as

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{Z}\mathbf{v} \tag{3.1}$$

In fact $\mathbf{Z}\mathbf{v}$ acts as feasible directions for matrix $\mathbf{A}$ (note that $\mathbf{A}\mathbf{Z}\mathbf{v} = \mathbf{0}$).

Now consider the following optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \quad \texttt{s.t.:} \quad \mathbf{A}^T\mathbf{x} = \mathbf{b} \tag{LEP}$$

using (3.1), it is possible to convert the linearly constrained optimization problem (LEP) on $\mathbb{R}^n$ to the following unconstrained optimization problem on the reduced space $\mathbb{R}^{(n-m)}$,

$$\min_{\mathbf{v}\in\mathcal{N}(\mathbf{A})} f(\mathbf{x}_0 + \mathbf{Z}\mathbf{v}) \tag{RLEP}$$

the gradient and Hessian of f with respect to the reduced vector $\mathbf{v}$ at the trial point $\mathbf{v}_0$ can be easily computed using the chain rule, i.e.,

$$\begin{aligned} \nabla_{\mathbf{v}} f(\mathbf{x}_0 + \mathbf{Z}\mathbf{v}_0) =& \mathbf{Z}^T \nabla_{\mathbf{x}} f(\mathbf{x}_0 + \mathbf{Z}\mathbf{v}_0) \\ \nabla^2_{\mathbf{v}} f(\mathbf{x}_0 + \mathbf{Z}\mathbf{v}_0) =& \mathbf{Z}^T \nabla^2_{\mathbf{x}} f(\mathbf{x}_0 + \mathbf{Z}\mathbf{v}_0)\mathbf{Z} \end{aligned}$$

the necessary and sufficient conditions for the reduced problem (RLEP) is same as the classical unconstrained optimization problems in which the gradient vector and the Hessian matrix are replaced by the reduced counterparts. Therefore, any unconstrained optimization solver can be employed to solve (RLEP) without any technical difficulty.

Now, consider the following inequality constrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \quad \texttt{s.t.:} \quad \mathbf{A}^T\mathbf{x} \geqslant \mathbf{b} \tag{LIP}$$

Using an appropriate active set strategy, it is possible to solve (LIP) with the null-space method. Assume the set of active constraints at the local solution is known, $\hat{\mathbf{A}}$, then the corresponding unconstrained optimization problem is solved on a null-space spanned by only active constraints, $\mathcal{N}(\hat{\mathbf{A}})$. In this case the caution should be taken for inactive constraints. Assume the index set of inactive constraints at $\mathbf{x}$ is shown by $\mathcal{I}$ (row-wise index). Denoting the search direction at $\mathbf{x}$ by $\mathbf{p} = \mathbf{Z}\mathbf{v}$, for all index $i \in \mathcal{I}$ so that $\mathbf{a}_i^T\mathbf{p} \geqslant 0$, any positive move along $\mathbf{p}$ will not violate the corresponding constraint ($\mathbf{a}_i$ denotes the i-th row of $\mathbf{A}$). Therefore, constraints with non-negative $\mathbf{a}_i^T\mathbf{p}$ do not pose any restriction on the stepsize. However, there is a critical step length, γ_i, for indices with $\mathbf{a}_i^T\mathbf{p} < 0$, where the constraint becomes binding, i.e., $\mathbf{a}_i^T(\mathbf{x}+\gamma_i\mathbf{p}) = \mathbf{b}_i$. So, the upper bound on the stepsize due to feasibility of iterates is computed by the following relation

$$\bar{\alpha} = \min\big\{ +\infty,\ \gamma_i = (\mathbf{b}_i - \mathbf{a}_i^T\mathbf{x})/(\mathbf{a}_i^T\mathbf{p}) \mid \mathbf{a}_i^T\mathbf{p} < 0\big\} \tag{3.2}$$

Similarly, the constrained optimization problem (LIP) can be converted to the following unconstrained optimization problem on the reduced space spanned by the active constraints

$$\text{(RLIP)} \qquad \min_{\mathbf{v}\in\mathcal{N}(\hat{\mathbf{A}})} f(\mathbf{x}_0 + \alpha\mathbf{Z}\mathbf{v}), \quad \alpha \in [0, \bar{\alpha}]$$

By some simple linear algebra, it is easy to show that

$$\texttt{cond}\big(\mathbf{Z}^T\nabla^2_{\mathbf{x}} f(\mathbf{x})\mathbf{Z}\big) \leqslant \texttt{cond}\big(\nabla^2_{\mathbf{x}} f(\mathbf{x})\big)\ \texttt{cond}\big(\mathbf{Z}\big)^2$$

therefore, to cope the possible instability during the computations (consider the limited precision arithmetic of computers), it is preferable to use an orthogonal null-space basis; $\texttt{cond}\big(\mathbf{Z}\big) = 1$. The QR factorization [Golub and Van Loan, 1996] is a common way to compute an orthogonal null-space for a desired full-rank matrix.

3.1. *QR factorization.* The QR factorization for the transpose of full-rank matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is expressed in the following form

$$\mathbf{A}^T = \mathbf{QR} = \begin{pmatrix} \mathbf{Q_1} & \mathbf{Q_2} \end{pmatrix} \begin{pmatrix} \mathbf{R_1} \\ \mathbf{0} \end{pmatrix}$$

where $\mathbf{Q}$ is an orthogonal matrix, $\mathbf{Q_1} \in \mathbb{R}^{n\times m}$, $\mathbf{Q_2} \in \mathbb{R}^{n\times(n-m)}$, $\mathbf{R_1} \in \mathbb{R}^{m\times m}$. Since $\mathbf{Q}$ is orthogonal, it follows that $\mathbf{AQ} = \mathbf{R}^T$, or $\mathbf{AQ_1} = \mathbf{R_1}^T$ and $\mathbf{AQ_2} = \mathbf{0}$ which results $\mathbf{Z} = \mathbf{Q_2}$.

In the present study, the Householder QR algorithm (cf. [Golub and Van Loan, 1996]) is used to compute the QR factorization of $\mathbf{A}$. In this method the orthogonal matrix $\mathbf{Q}$ is represented by the product of Householder reflection matrixes

$$\mathbf{Q} = \mathbf{H_1}\mathbf{H_2}\ldots\mathbf{H_m}$$

where every $\mathbf{H_i} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix with the following generic form

$$\mathbf{H} = \mathbf{I} - \tau \mathbf{u}\mathbf{u}^T$$

where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix, $\tau \in \mathbb{R}$ and $\mathbf{u} \in \mathbb{R}^n$. For details of computing $\mathbf{H_i}$ factors refer to [Golub and Van Loan, 1996]. In the following we adapt the Householder QR factorization algorithm to compute an orthogonal null-space for linear constraint in knapsack problems.

In the knapsack problems, we have at most one active linear constraint, so we need to compute the QR factorization for $\mathbf{A} = \mathbf{a}$. Without loss of generality, assume that $a_1 \neq 0$ (else a simple pivoting should be applied). In this case the factor $\mathbf{Q}$ can be represented in the following form

$$\mathbf{Q} = \mathbf{I} - \tau \mathbf{u}\mathbf{u}^T$$

where

$$u_1 = 1, \quad u_i = \frac{a_i}{a_1 - \zeta}, \quad i = 2, \ldots, n, \quad \tau = \frac{\zeta - a_1}{\zeta}, \quad \zeta = -\mathtt{sign}(a_1)\ \|\mathbf{a}\|_2$$

moreover, the matrix $\mathbf{R} = \mathbf{R_1}$ is a 1×1 matrix such that its singleton entry is equal to ζ. The following piece of code shows the C implementation of the mentioned factorization algorithm; in which `a`, `tau` and `u` are respectively identical to $\mathbf{a}$, τ and $\mathbf{u}$ in the above discussion.

```c
void factorize (int n, double *a, double *tau, double *u) {
  int i; double zeta, gamma;
  for(i = 0, zeta = 0.0; i < n; zeta+ = a[i] * a[i++]);
  zeta = (a[0] < 0.0) ? sqrt(zeta) : -sqrt(zeta);
  *tau = (zeta - a[0]) / zeta;
  gamma = 1.0 / (a[0] - zeta);
  for(i = 1, u[1] = 1.0; i < n; u[i] = gamma * a[i++]);
}
```

In the null-space method we frequently need to do product of null-space matrix, $\mathbf{Z}$, or its inverse, $\mathbf{Z}^T$ (consider orthogonality of $\mathbf{Z}$), with a desired vector. Notice that matrix $\mathbf{Z}$ is formed by removing the first column of matrix $\mathbf{Q}$. Exploiting the specific representation of $\mathbf{Q}$ it is possible to perform the product of $\mathbf{Z}$ with an arbitrary vector $\mathbf{v} \in \mathbb{R}^{n-1}$ in O(n) arithmetic operations. Considering a pseudo entry $v_0 = 0$ for vector $\mathbf{v}$, simple linear algebra results

$$\mathbf{zv} = \mathbf{Zv}, \quad zv_i = v_{i-1} - \tau u_i \sum_{j=1}^{n-1} (u_{j+1} v_j), \quad i = 1, \ldots, n.$$

where $\mathbf{zv} \in \mathbb{R}^n$ is expansion of the reduced vector $\mathbf{v} \in \mathbb{R}^{n-1}$ to the full-space. The following piece of code shows the C implementation of this product; in which `tau`, `u`, `v` and `zv` are respectively identical to τ, $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{zv}$.

```c
void multZ (int n, double tau, double *u, double *v,
             double *zv) {
  int i; double t = zv[0] = 0.0;
  for(i = 0; i < n-1; zv[i+1] = v[i], t += u[i+1] * v[i++]);
  t *= tau;
  for(i = 1, zv[0] -= t; i < n; zv[i] -= t * u[i++]);
}
```

In the similar way, product of $\mathbf{Z}^T$ with an arbitrary vector $\mathbf{w} \in \mathbb{R}^n$ can be computed by the following relation

$$\mathbf{ztw} = \mathbf{Z}^T\mathbf{w}, \quad ztw_i = w_{i+1} - \tau u_{i+1}\sum_{j=1}^{n}(u_i w_i), \quad i = 1, \dots, n-1.$$

The C implementation of this product is shown in the following piece of code:

```c
void multZt (int n, double tau, double *u, double *w,
              double *ztw) {
  int i; double t = w[0];
  for(i = 1; i < n; t += u[i] * w[i++]);
  t *= tau;
  for(i = 0; i < n-1; ztw[i] = w[i+1] - t * u[i+1], i++);
}
```

3.2. *An alternative null-space by orthogonal projection matrix.* The orthogonal projection matrix is another possible choice for the null-space of full rank matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$. The orthogonal projection matrix, $\mathbf{P} \in \mathbb{R}^{m\times n}$, can be computed by the following relation

$$\mathbf{P} = \mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}$$

The main difference of this null-space basis with the previous mentioned one is that the orthogonal projection matrix is not full rank. Therefore, the equivalent unconstrained optimization problem will be solved on the full-space $\mathbb{R}^n$. This may simplifies the implementation of method; but probably; in expense of more computational cost.

Notice that the name ”orthogonal projection matrix” should not make misleading about the properties of this null-space basis, as in general $\mathbf{P}$ is not an orthogonal matrix. Therefore, the stability of computation can be in question in using $\mathbf{P}$ as the null-space matrix.

Without regard to the stability of computation, the special structure of $\mathbf{A}$ in the knapsack problems, makes the orthogonal projection matrix an attractive choice to make a null-space basis. In this case $\mathbf{P}$ can be computed and stored explicitly; without any cost; as follows

$$\mathbf{P} = \mathbf{I} - \left(\mathbf{a}^T\mathbf{a}\right)^{-1} \mathbf{a}\mathbf{a}^T$$

The product of $\mathbf{P}$ with an arbitrary vector $\mathbf{v} \in \mathbb{R}^n$ can be computed within O(n) operations; as follows

$$\mathbf{pv} = \mathbf{P}\mathbf{v}, \quad pv_i = v_i - a_i \,\frac{\sum_{j=1}^{n}(a_i v_i)}{\sum_{j=1}^{n}(a_i a_i)}, \quad i = 1, \ldots, n$$

where $\mathbf{pv} \in \mathbb{R}^n$ is the projection of vector $\mathbf{v}$ onto the null-space of $\mathbf{A}$. Using theorem 2.2, it is trivial to show that the orthogonal projection of an arbitrary point $\mathbf{y} \in \mathbb{R}^n$ onto equality constraint of knapsack problem (ignoring bound constraints) which is denoted by $\mathbf{z} \in \mathbb{R}^n$ can be computed by

$$z_i = y_i - a_i \,\frac{b - \sum_{j=1}^{n}(a_j y_j)}{\sum_{j=1}^{n} a_j^2}, \quad i = 1, \ldots, n \tag{3.3}$$

In the similar way, using theorem 2.7, projection point onto knapsack bilateral inequality constraint (ignoring bound constraints) is computed by the following relation

$$\mathbf{z} = \mathtt{mid}(\mathbf{zl},\ \mathbf{y},\ \mathbf{zu}) \tag{3.4}$$

where $\mathbf{zl}$ and $\mathbf{zu}$ are computed by (3.3) in which b is replaced by b_l and b_u respectively. Equations (3.3) and (3.4) are useful for treatment of linear constraints by projection in the unconstrained solver.

3.3. *Unconstrained optimization on the reduced space.* In this subsection we want to comment on the exploiting an unconstrained optimization solver on the null-space of (active) linear constraints. Basically this procedure is straightforward and the unconstrained optimization solver can be blind about the constrained problem, i.e., we can use the unconstrained optimization solver as a black-box. The following items are sufficient to exploit the unconstrained solver for this purpose:

(a) At the initial step we need the starting point $\mathbf{x}_0 \in \mathbb{R}^n$ which is feasible with respect to linear constraints. It can be achieved by projecting a trail point onto the constraint set. This point is stored in memory and used during the unconstrained optimization as described below. The starting point of unconstrained optimization solver, $\mathbf{v}_0 \in \mathbb{R}^{n-m}$, can be taken equal to $\mathbf{0} \in \mathbb{R}^{n-m}$.

(b) When the unconstrained solver asks for the value of the objective function at the trial point $\mathbf{v}_k \in \mathbb{R}^{n-m}$, we should first compute the corresponding value of $\mathbf{v}_k$ on the full-space, $\mathbf{x}_k \in \mathbb{R}^n$, by this relation $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{Z}\mathbf{v}_k$. Then $f(\mathbf{x}_k)$ is passed to the unconstrained solver as the value of the objective function at $\mathbf{v}_k$.

(c) Similar to item (b), when the unconstrained solver asks for the gradient of the objective function, $\nabla_{\mathbf{v}} f(\mathbf{v}_k)$, at the trial point $\mathbf{v}_k$, we first compute $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{Z}\mathbf{v}_k$. Then pass $\mathbf{Z}^T \nabla_{\mathbf{x}} f(\mathbf{x}_k)$ to the unconstrained solver as the desired gradient.

(c) Similarly, when the unconstrained solver asks for the Hessian matrix of the objective function, $\nabla^2_{\mathbf{v}} f(\mathbf{v}_k)$, at the trial point $\mathbf{v}_k$, the value of $\mathbf{Z}^T \nabla^2_{\mathbf{x}} f(\mathbf{x}_0 + \mathbf{Z}\mathbf{v}_k)\mathbf{Z}$ is passed to it as the desired Hessian matrix.

(d) The stepsize related to the globalization strategy of the unconstrained solver should be restricted to interval $[0, \bar{\alpha}]$, where $\bar{\alpha}$ is computed by (3.2). In the present study, this item only applied in the case of inequality knapsack problems. If the active set of linear constraints at the optimal solution be determined in a finite number of iterations, this stepsize restriction does not destroy the local convergence rate of the unconstrained solver. For the finite-cycle determination of active set, it is required that the new active set be a sub-set of the previous one; which is possible to ensure in the case of no non-degeneracy of the optimal solution with respect to the linear active constraints (the lagrange multiplier corresponding to the active constraint be non-zero at the local solution).

4. Hager-Zhang active set algorithm for knapsack problems.

Consider a bound constrained nonlinear program, if the set of active constraints at a local solution be a-priori known, it is possible to fix these constraints to the corresponding bound values and solve an unconstrained optimization problem on the reduced space spanned by the free variables. This is the key idea of the active set strategy. In practice, the set of active indices in not a-priori known, therefore they should be identified by an appropriate prediction correction strategy. Under appropriate conditions, this procedure can be performed in a finite number of iterations. However, in general it is possible to add (remove) only one index to (from) the current active set. This increases the necessary number of iterations in particular for large scale problems. Fortunately, it is possible to add many constraints to working set by means of the projected gradient method. This idea was used in [Lin and Moré, 1999; Heinkenschloss et al., 1999; Birgin and Martínez, 2002; Hager and Zhang, 2006] to efficiently exploit the active set strategy.

In this section we adapt the Hager-Zhang active set algorithm (HZ-ASA) [Hager and Zhang, 2006] to solve knapsack problems. Using the previously constructed tools, this extension is trivial and the convergence theory holds almost under the same conditions. The main reason for selection of this algorithm is its excellent convergence theories in addition to the promising numerical results reported in [Hager and Zhang, 2006]. Moreover, unlike [Lin and Moré, 1999; Heinkenschloss et al., 1999], this method admits the superlinear convergence under the same conditions while it does not need any explict form of second order information and/or solution of system of linear equations per cycle. These properties makes it an ideal choice to solve very large scale problems. The current author believes that the key efficiency of this method can be connected to its two-phase nature. Unlike the related methods which use the same interactions in the course of optimization, this method start with a cheap constrained first-order method and after sufficient progress toward a local solution, branches to a (more expensive) higher-order unconstrained solver. In other words, it does not waste the energy at early stages by performing expensive and accurate steps. Under certain conditions, the method only branches once between two phases. This strategy is particularly effective when the initial guess is poor. Note that the basic idea for this strategy is not new and already used in [Birgin and Martínez, 2002].

In [Hager and Zhang, 2006] the nonmonotone spectral projected gradient [Birgin et al., 2000] (SPG) was used to identify the working set. In this method, the search direction is parallel to the steepest descent direction (is descent) and the stepsize is computed by projecting the (spectral) Barzilai-Borwein [Barzilai and Borwein, 1988] step onto the feasible set. Moreover, the Grippo-Lampariello-Lucidi nonmonotone line search algorithm [Grippo et al., 1986] is used to ensure the convergence to a local minimum from an arbitrary initial guess, simultaneously benefiting from the spectral property of the Barzilai-Borwein stepsize as much as possible. The SPG method can be efficiently applied to every convex constrained optimization problem (under some common conditions) providing an efficient method to project a trial point onto the feasible set. Therefore, by means of O(n) projection algorithm introduced in section 2, SPG can be effectively used to solve large scale knapsack problems. However, the local convergence of SPG method will not be better than linear in the vicinity of the local solution.

Let $\alpha \in \mathbb{R}$, the scaled projected gradient, $\mathbf{d}^{\alpha}(\mathbf{x})$ is defined as follows

$$\mathbf{d}^{\alpha}(\mathbf{x}) = \mathcal{P}_{\mathcal{D}}(\mathbf{x} - \alpha \nabla_{\mathbf{x}} f(\mathbf{x})^T) - \mathbf{x}$$

It is easy to show that $\mathbf{d}^{\alpha}(\mathbf{x})$ is a constrained descent direction (e.g. see: lemma 2.1 of [Birgin et al., 2000] or proposition 2.1 of [Hager and Zhang, 2005]). Therefore, it can be used as the stopping criteria to measure the distance of current iterate from the optimal solution. Assume $\mathbf{s}_k = (\mathbf{x}_k - \mathbf{x}_{k-1})$ and $\mathbf{y}_k = (\nabla_{\mathbf{x}} f(\mathbf{x}_k) - \nabla_{\mathbf{x}} f(\mathbf{x}_{k-1}))$. In the Barzilai-Borwein method, the trial stepsize, α_k, is computed from a one-dimensional search method based on the following relation

$$\alpha_k = \arg\min_{\alpha \in \mathbb{R}} \frac{1}{2} \, \|\mathbf{D}(\alpha)\ \mathbf{s}_{k-1} - \mathbf{y}_{k-1}\|^2 = (\mathbf{s}_{k-1}^T \mathbf{s}_{k-1})/(\mathbf{s}_{k-1}^T \mathbf{y}_{k-1}). \tag{4.1}$$

where $\mathbf{D}(\alpha) = \frac{1}{\alpha}\mathbf{I}$ is a diagonal approximation to the Hessian matrix $\nabla^2_{\mathbf{x}} f(\mathbf{x})$. The spectral property of Barzilai-Borwein method can be inferred from the following lemma:

LEMMA 4.1. *The Barzilai-Borwein stepsize, is equal to inverse of Rayleigh quotient, related to vector s_{k-1}, of the averaged Hessian of the objective functional between two consecutive iterations $k-1$ and k.*

PROOF. Using the Mean-Value Theorem it is easy to show that:

$$(\nabla_{\mathbf{x}} f(\mathbf{x}_k) - \nabla_{\mathbf{x}} f(\mathbf{x}_{k-1}))/(\mathbf{x}_k - \mathbf{x}_{k-1}) = \int_0^1 \nabla^2 f(t\mathbf{x}_k + [1-t]\mathbf{x}_{k-1})dt,$$

therefore,

$$(\mathbf{s}_{k-1}^T \mathbf{y}_{k-1})/(\mathbf{s}_{k-1}^T \mathbf{s}_{k-1}) = \left(\mathbf{s}_{k-1}^T \Big(\int_0^1 \nabla^2 f(\mathbf{x}_{k-1} + t\mathbf{s}_{k-1})dt\Big)\mathbf{s}_{k-1}\right)/(\mathbf{s}_{k-1}^T \mathbf{s}_{k-1}),$$

which complete the proof. $\square$

By lemma 4.1, $\Lambda_{max}^{-1} \leqslant \alpha_k \leqslant \Lambda_{min}^{-1}$, where Λ_{max} and Λ_{min} are the maximum and minimum eigenvalues of the averaged Hessian matrix respectively. Therefore, $\alpha_k \mathbf{I}$ is a consistent approximation to the inverse of the Hessian matrix. The statement of SPG algorithm is as follows:

Algorithm 1: Outline of spectral projected gradient (SPG) algorithm

```
1 given $\epsilon \in [0, \infty)$, $0 < \sigma_{min} < \sigma_{max} < \infty$
2 take $k = 0$ and $\mathbf{x}_0 \in \mathcal{D}$
3 while $\|\mathbf{d}_k^1\| > \epsilon$ do
4     if $(k = 0$ or $\mathbf{s}_k^T \mathbf{y}_k \leqslant 0)$ then $\sigma = 1$
5     else $\sigma = \mathtt{mid}(\sigma_{min}, \frac{\mathbf{s}_k^T \mathbf{s}_k}{\mathbf{s}_k^T \mathbf{y}_k}, \sigma_{max})$
6     take $\mathbf{d}_k^\sigma$ as the search direction
7     compute stepsize $\alpha_k$ by nonmonotone line search
8     $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k^\sigma$, $k = k + 1$
9 end
```

Note that the condition $\mathbf{s}_k^T \mathbf{y}_k \leqslant 0$ in line 4 of algorithm 1 means the detection of negative curvature directions. Assuming that the objective function is consistent with its quadratic approximation around the current iterate, the large stepsize $\sigma = 1$ is used to achieve the maximum reduction. The statement of the nonmonotone line search used in this study is as follows:

Algorithm 2: Outline of nonmonotone line search algorithm

```
1 given $M \in \mathbb{N}$ $(M > 0)$, $\gamma \in (0, 1)$, $0 < \alpha_{min} < \alpha_{max} < 1$,
2 take $\alpha = 1$, $\delta = (\mathbf{d}_k^\sigma)^T \nabla_{\mathbf{x}} f_k$, $\mathbf{x}_+ = \mathbf{x}_k + \mathbf{d}_k^\sigma$
3 $f_k^{max} = \max\{f_{k-j} \mid 0 \leqslant j \leqslant \min\{k, M-1\}\}$
4 while $(f(\mathbf{x}_+) > f_k^{max} + \alpha\gamma\delta)$ do
5     $\alpha_t = \frac{1}{2}\,\alpha^2 \delta / (f(\mathbf{x}_+) - f(\mathbf{x}_k) - \alpha\delta)$
6     if $(\alpha_{min} \leqslant \alpha_t \leqslant \alpha\alpha_{max})$ then $\alpha = \alpha_t$
7     else $\alpha = \alpha/2$
8     $\mathbf{x}_+ = \mathbf{x}_k + \alpha \mathbf{d}_k^\sigma$
9 end
```

Notice that when the decrease condition in line 4 of algorithm 2 is not satisfied a quadratic interpolation (line 5) is used to compute the trail step α_t and it is accepted if lies within interval $[\alpha_{min}, \alpha\alpha_{max}]$, otherwise a bisection will be performed (line 7). The quadratic function $q(\alpha)$ is formed such that $q(0) = f(\mathbf{x}_k)$, $q(\alpha) = f(\mathbf{x}_+)$ and $dq/d\alpha = \delta$.

Theorem 4.2. *([Birgin et al., 2000] theorem 2.4) Algorithm SPG is well-defined, and any accumulation point of the sequence $\{\mathbf{x}_k\}$ that it generates is a constrained stationary point.*

Further details about SPG algorithm and its convergence theory can be found in either of [Birgin et al., 2000] or [section 2: Hager and Zhang, 2006].

The unconstrained solver (US) used in HZ-ASA is the CGDESCENT algorithm [Hager and Zhang, 2005] which possesses a superlinear local convergence rate under some common conditions. In HZ-ASA the unconstrained solver is modified to be enable to manage infeasible iterations (with respect to bound constraints). Similar to (3.2), it includes the modification of the stepsize such that the new iterates will not be infeasible; i.e., the trial stepsize is limited to the interval $[0, \hat{\alpha}]$ where,

$$\hat{\alpha} = \max\{\alpha \in \mathbb{R}^+ \mid (\mathbf{x}_I(\mathbf{x}) + \alpha \mathbf{p}_I(\mathbf{x})) \in \mathcal{B}\} \tag{4.2}$$

The outline of the reduced CGDESCENT (RCGD) algorithm used in this study is as follows:

Algorithm 3: Outline of RCGD algorithm

```
1  given $\epsilon \in [0,\infty)$, $0 < \alpha_{min} < \alpha_{max} < \infty \mathbf{x}_0 \in \mathbb{R}^n$, $\mathbf{x}_0 \in \mathcal{D}$, $t = \mathtt{dim}\ (\mathcal{A}_0)$
2  take $\mathbf{v}_0 = 0$, $\mathbf{d}_0 = \mathbf{g}_0 = (\mathbf{Z}^T\mathtt{shrink}(\nabla_{\mathbf{x}} f(\mathbf{x}_0)))^T$, $k = 0$
3  while $\|\mathbf{g}_k\| > \epsilon$ do
4      find $\alpha_k$ by monotone line search
5      compute $\bar{\alpha}_k$ and $\hat{\alpha}_k$ using (3.2) and (4.2) respectively
6      $\tilde{\alpha}_k = \min\{\alpha_k, \bar{\alpha}_k, \hat{\alpha}_k\}$, $\tilde{\alpha}_k = \mathtt{mid}(\alpha_{min}, \tilde{\alpha}_k, \alpha_{max})$
7      $\mathbf{v}_{k+1} = \mathbf{v}_k + \tilde{\alpha}_k \mathbf{d}_k$
8      $\mathbf{g}_{k+1} = (\mathbf{Z}^T\mathtt{shrink}(\nabla_{\mathbf{x}} f(\mathbf{x}_0 + \mathtt{expand}(\mathbf{Z}\mathbf{v}_{k+1}))))^T$
9      $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$
10     $\beta_k = \frac{\mathbf{g}_{k+1}^T}{\mathbf{d}_k^T\mathbf{y}_k}(\mathbf{y}_k - 2\mathbf{d}_k\frac{\|\mathbf{y}_k\|^2}{\mathbf{d}_k^T\mathbf{y}_k})$
11     $\mathbf{d}_{k+1} = \beta_k \mathbf{d}_k - \mathbf{g}_{k+1}$, $k = k+1$
12 end
```

Notice that the value of active constraints will be fixed during RCGD iterations. The feasibility of free indices with respect to bound constraints will be maintained by $\hat{\alpha}_k$. When $\mathcal{D} \overset{def}{=} \mathcal{D}_{\mathcal{E}}$ then we have always one active linear constraint and work with $\mathbf{d}$, $\mathbf{g}$, $\mathbf{v}$, $\mathbf{y} \in \mathbb{R}^{n-t-1}$ and $\bar{\alpha}_k = \infty$. When $\mathcal{D} \overset{def}{=} \mathcal{D}_{\mathcal{I}}$ and both of the linear inequality constraints be inactive $\mathbf{d}$, $\mathbf{g}$, $\mathbf{v}$, $\mathbf{y} \in \mathbb{R}^{n-t}$ and $\mathbf{Z} = \mathbf{I}$. In this case the feasibility of iterations with respect to equality constraints will be controlled by $\bar{\alpha}_k$. Finally, when $\mathcal{D} \overset{def}{=} \mathcal{D}_{\mathcal{I}}$ and one of the linear inequality constraints be active we will work with $\mathbf{d}$, $\mathbf{g}$, $\mathbf{v}$, $\mathbf{y} \in \mathbb{R}^{n-t-1}$ and the feasibility of iterations with respect to linear constraints will be controlled by $\bar{\alpha}_k$.

The monotone line-search used in RCGD algorithm should satisfy either of the standard Wolfe conditions or approximate Wolfe conditions introduced in [Hager and Zhang, 2005]. In [Hager and Zhang, 2005] a robust line search algorithm is suggested which tolerates the finite accuracy precision of computer arithmetic very well. The global convergence and superlinearly local convergence of the presented RCGD algorithm can be directly followed from the convergence theories of the original CGDESCENT algorithm discussed in [Hager and Zhang, 2005].

Following [Hager and Zhang, 2006], the set of undecided indices is defined as follows

$$\mathcal{U}(\mathbf{x}) = \{i \in [1, n] \mid g_i(\mathbf{x}) \geqslant \|\mathbf{d}^1(\mathbf{x})\|^a \texttt{ and } \min\{x_i - l_i, u_i - x_i\} \geqslant \|\mathbf{d}^1(\mathbf{x})\|^b\}$$

where $g_i(\mathbf{x})$ denotes i-th component of $\nabla_{\mathbf{x}} f(\mathbf{x})$, $a \in (0, 1)$ and $b \in (1, 2)$ (e.g. $a = 1/2$ and $b = 3/2$). In fact, $\mathcal{U}(\mathbf{x})$ denotes the set of indices correspond to components of $\mathbf{x}$ for which the associated gradient component, $g_i(\mathbf{x})$, is relatively large while x_i is not close to either of l_i or u_i. When $\mathcal{U}(\mathbf{x})$ is empty, it implies that the indices with large associated gradient component are almost identified and it is preferable to perform unconstrained optimization algorithm on the reduced space of free indices. This naturally happens in the vicinity of the local solution.

Now let us to recall the HZ-ASA [Hager and Zhang, 2006] here. Using the above mentioned SPG and RCGD algorithm it is straightforward to use HZ-ASA algorithm to solve knapsack problems. The outline of HZ-ASA for knapsack problems is as follows:

Algorithm 4: Outline of HZ-ASA for knapsack problems

```
1  $\epsilon \in [0, \infty)$, $\mu$, $\rho \in (0, 1)$, $n_1$, $n_2 \in [1, n)$
   **Phase 1 (Active set identification by SPG)**:
2  while $\|\mathbf{d}_k^1\| > \epsilon$ do
3      Do a SPG iteration and check the following:
4      if $\mathcal{U}(\mathbf{x}_k) = \emptyset$ then
5          if $\|\mathtt{shrink}(\nabla_{\mathbf{x}} f(\mathbf{x}_k))\| > \mu\|\mathbf{d}_k^1\|$ then $\mu = \rho\mu$
6          else goto Phase 2
7      else
8          if $\mathcal{A}_k = \mathcal{A}_{k-1} = \ldots = \mathcal{A}_{k-n_1}$ and $\|\mathtt{shrink}(\nabla_{\mathbf{x}} f(\mathbf{x}_k))\| \geqslant \mu\|\mathbf{d}_k^1\|$
           then goto Phase 2
9      end
10 end
   **Phase 2 (Unconstrained optimization by RCGD)**:
11 while $\|\mathbf{d}_k^1\| > \epsilon$ do
12     Do a RCGD iteration and check the following:
13     if $\|\mathtt{shrink}(\nabla_{\mathbf{x}} f(\mathbf{x}_k))\| > \mu\|\mathbf{d}_k^1\|$ then goto Phase 1 and restart SPG
14     if $|\mathcal{A}_{k-1}| < |\mathcal{A}_k|$ then
15         if $\mathcal{U}(\mathbf{x}_k) = \emptyset$ or $|\mathcal{A}_k| > |\mathcal{A}_{k-1}| + n_2$ then restart RCGD at $\mathbf{x}_k$
16         else goto Phase 1 and restart SPG
17     end
18 end
```

The convergence theories stated in [Hager and Zhang, 2006] for HZ-ASA is almost hold for algorithm 4. In the remaining part of this section, we shall re-state results of [Hager and Zhang, 2006] for algorithm 4.

THEOREM 4.3. *(global convergence) Let $\mathcal{L}$ be the level set defined by*

$$\mathcal{L} = \{\mathbf{x} \in \mathcal{D} : f(\mathbf{x}) \leq f(\mathbf{x}_0)\ \}.$$

We assume the following conditions hold:

G1. *f is bounded from below in $\mathcal{L}$ and $d_{\max} = \sup_k \|\mathbf{d}_k\| < \infty$.*

G2. *If $\bar{\mathcal{L}}$ is the collection of $\mathbf{x} \in \mathcal{D}$ whose distance to $\mathcal{L}$ is at most $d_{\max}$, then ∇f is Lipschitz continuous on $\bar{\mathcal{L}}$.*

Then either algorithm 4 with $\epsilon = 0$ terminates in a finite number of iterations at a stationary point, or we have $\liminf_{k\to\infty} \|\boldsymbol{d}^1(\boldsymbol{x}_k)\|_\infty = 0$.

Proof. followed from theorem 4.1 of [Hager and Zhang, 2006]. □

Theorem 4.4. *(global minimizer) If f is strongly convex and twice continuously differentiable on $\mathcal{D}$, then the iterates $\mathbf{x}_k$ of algorithm 4 with $\epsilon = 0$ converge to the global minimizer of (1.1).*

Proof. see corollary 4.2 of [Hager and Zhang, 2006]. □

The following theorem results the local superlinear convergence rate of algorithm 4 to a nondegenerate stationary point.

Theorem 4.5. *(nondegenerate local convergence) If f is continuously differentiable and the iterates $\mathbf{x}_k$ generated by algorithm 4 with $\epsilon = 0$ converge to a nondegenerate (with respect to both of the bound and linear constraints) stationary point $\mathbf{x}^*$, then after a finite number of iterations, algorithm 4 performs only RCGD iterations without restarts.*

Proof. followed from theorem 5.1 of [Hager and Zhang, 2006]. □

The following theorems results the local superlinear convergence rate of algorithm 4 to a stationary point (even degenerate stationary point) under strong second-order sufficient optimality condition.

Theorem 4.6. *(degenerate local convergence) Assume $\mathcal{D} \overset{def}{=} \mathcal{D}_\mathcal{E}$ in problem (1.1) and f is is twice-continuously differentiable. If the iterates $\mathbf{x}_k$ generated by algorithm 4 with $\epsilon = 0$ converge to a stationary point $\mathbf{x}^*$ satisfying the strong second-order sufficient optimality condition, then after a finite number of iterations, algorithm 4 performs only RCGD iterations without restarts.*

Proof. followed from theorem 5.7 of [Hager and Zhang, 2006]. □

Theorem 4.7. *(degenerate local convergence) Assume $\mathcal{D} \overset{def}{=} \mathcal{D}_\mathcal{I}$ in problem (1.1) and f is is twice-continuously differentiable. If the iterates $\mathbf{x}_k$ generated by algorithm 4 with $\epsilon = 0$ converge to a stationary point $\mathbf{x}^*$ satisfying* *the strong second-order sufficient optimality condition and* $\mathbf{x}^*$ *is not degenerate with respect to the linear inequality constraint, i.e., the optimal value of lagrange multiplier corresponding to the active linear constraint is nonzero, then after a finite number of iterations, algorithm 4 performs only RCGD iterations without restarts.*

Proof. followed from theorem 5.7 of [Hager and Zhang, 2006]. □

Remark 4.8. There is a simple strategy to virtually benefit from the local superlinear convergence rate of algorithm 4 in the case of degeneracy of the minimizer with respect to linear inequality constraints. Since the different states of the active set of (1.1) with respect to linear constraints includes only three situtions, it is possible to solve (1.1) sequentially three times with algorithm 4 using its strong local convergence. Assume, $\mathbf{x}_m^*$, $\mathbf{x}_l^*$ and $\mathbf{x}_u^*$ are solutions of (1.1) by algorithm 4 without considering the linear constraint in RCGD, considering the linear equality constraint with $b = b_l$ in RCGD and considering the linear equality constraint with $b = b_u$ in RCGD respectively. Note that we can first solve for $\mathbf{x}_m^*$ to possibly avoid further computation. If $\mathbf{x}_m^*$ satisfies the bilateral inequality constraint, it is equal to a local solution of problem and the computation will be discarded (of course in this case there is no degeneracy with respect to linear constraints). Otherwise, the local solution of problem is one of $\mathbf{x}_l^*$ or $\mathbf{x}_u^*$ which correspond to the smaller objective function value. Therefore, this strategy finds a local solution of problem by solving three sequential superlinearly convergent optimizations. However, as the cost of computation is tripled by this strategy, the overall rate of convergence can be sub-linear in the worst conditions.

The computer implementation of the presented algorithm together with some numerical experiments can be downloaded from the following URL:

http://mehr.sharif.ir/∼tav/asak.html

5. Summary.

In the present study, a general framework is suggested to adapt bound-constrained optimization solvers to solve optimization problems with bounds and an additional linear constraint (in the form of either equality or bilateral inequality). This framework includes two main ingredients which are the projection of an arbitrary point onto the feasible set and null-space manipulation of the related linear constraint. The implementation of the method with specific focus on the Hager-Zhang active set algorithm Hager and Zhang [2006] is discussed in details.

It seems that, following the presented approach, it should be possible to perform such an extension for alternative box-constrained optimization methods. Notice that this extension can be much simpler and straightforward for some box-constrained solvers. For instance, using just the projection operator introduced in this study, it is natural to extend Newton method of Lin and Moré [1999] or affine-scaling interior-point Newton method of Heinkenschloss et al. [1999] for this purpose without further difficulties.

Acknowledgements. The authors would like to thanks Marina Andretta and Ernesto Birgin for constructing comments.

References.

J. Barzilai and J.M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

E.G. Birgin and J.M. Martínez. Large-scale active-set box-constrained optimization method with spectral projected gradients. *Computational Optimization and Applications*, 23(1):101–125, 2002.

E.G. Birgin, J.M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

R.P. Brent. *Algorithms for minimization without derivatives.* Courier Dover Publications, 2002.

K.M. Bretthauer and B. Shetty. The nonlinear knapsack problem–algorithms and applications. *European Journal of Operational Research*, 138(3):459–472, 2002.

W. J. Cody. Algorithm 665: Machar: a subroutine to dynamically determined machine parameters. *ACM Trans. Math. Softw.*, 14(4):303–311, 1988. ISSN 0098-3500. .

K. Dahiya, S.K. Suneja, and V. Verma. Convex programming with single separable constraint and bounded variables. *Computational Optimization and Applications*, 36(1): 67–82, 2007.

Y.H. Dai and R. Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100(1):21–47, 2005.

Y.H. Dai and R. Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming*, 106(3):403–421, 2006.

F. Facchinei, S. Lucidi, and L. Palagi. A Truncated Newton Algorithm for Large Scale Box Constrained Optimization. *SIAM Journal on Optimization*, 12(4):1100–1125, 2002.

P.E. Gill, W. Murray, and M.H. Wright. *Practical optimization.* Academic Press London, 1981.

G.H. Golub and C.F. Van Loan. *Matrix computations.* Johns Hopkins Univ Pr, 1996.

M.D. Gonzalez-Lima, W.W. Hager, and H. Zhang. An Affine-scaling Interior-point Method for Continuous Knapsack Constraints. preprint, 25 July, 2009.

L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for Newton's method. *SIAM Journal on Numerical Analysis*, pages 707–716, 1986.

W.W. Hager and H. Zhang. A New Conjugate Gradient Method with Guaranteed Descent and an Efficient Line Search. *SIAM Journal on Optimization*, 16(1):170–192, 2005.

W.W. Hager and H. Zhang. A New Active Set Algorithm for Box Constrained Optimization. *SIAM Journal on Optimization*, 17(2):526–557, 2006.

W.W. Hager, B.A. Mair, and H. Zhang. An affine-scaling interior-point CBB method for box-constrained optimization. *Mathematical Programming*, 119(1):1–32, 2009.

M. Heinkenschloss, M. Ulbrich, and S. Ulbrich. Superlinear and quadratic convergence of affine-scaling interior-point Newton methods for problems with simple bounds without strict complementarity assumption. *Mathematical Programming*, 86(3):615–635, 1999.

K.C. Kiwiel. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Mathematical Programming*, 112(2):473–491, 2008.

C.J. Lin and J.J. Moré. Newton's Method for Large Bound-Constrained Optimization Problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999.

C.J. Lin, S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone. Decomposition Algorithm Model for Singly Linearly-Constrained Problems Subject to Lower and Upper Bounds. *Journal of Optimization Theory and Applications*, 141(1):107–126, 2009.

S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone. A convergent decomposition algorithm for support vector machines. *Computational Optimization and Applications*, 38(2):217–234, 2007.

A. Melman and G. Rabinowitz. An efficient method for a class of continuous nonlinear knapsack problems. *SIAM Review*, 42(3):440–448, 2000.

J.J. Moré and S.A. Vavasis. On the solution of concave knapsack problems. *Mathematical Programming*, 49(1):397–411, 1990.

M. Patriksson. A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research*, 185(1):1–46, 2008.

P. Tseng and S. Yun. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications*, In press, 2008.

J.H. Wilkinson. *Rounding errors in algebraic processes.* Dover Pubns, 1994.

Department of Materials Science
Sharif University of Technology
Tehran, Iran, P.O. Box 11365-9466
E-mail: tav@mehr.sharif.ir
URL: http://mehr.sharif.ir/∼tav/