A Trust Region Method for the Solution of the Surrogate Dual in Integer Programming

N. L. Boland[†], A. C. Eberhard[§] and A. Tsoukalas[§]

[†]Faculty of Science and Information Technology,
School of Mathematical and Physical Sciences,
University of Newcastle, University Drive,
Callaghan NSW 2308 Australia

[§]School of Mathematical and Geospatial Sciences, RMIT
GPO Box 2476V, Melbourne,
Victoria, Australia 3001.*

Abstract

We propose an algorithm for solving the surrogate dual of a mixed integer program. The algorithm uses a trust region method based on a piecewise affine model of the dual surrogate value function. A new and much more flexible way of updating bounds on the surrogate dual's value is proposed, which numerical experiments prove to be advantageous. A proof of convergence is given and numerical tests show that the method's performance is superior to a state-of-the-art subgradient solver.

1 Introduction

Consider the mixed integer program (MIP)

$$\begin{aligned} &\min cx \\ &\text{subject to } Ax \geq b \\ &\qquad x \in X \end{aligned} \tag{MIP}$$

where X contains the integrality constraints and other "simple constraints" such as bounds on variables. In the following we will denote products of vectors and matrices by their simple juxtaposition, where we assume the dimensions are adjusted so that multiplication is compatible i.e. xu assume x is $1 \times n$ and u is $n \times 1$ while ux assume u is $1 \times n$ and x is $n \times 1$. Given a problem (P) we denote its value by $v(P)$ and its minimizers (or maximizers) by $\arg\min$(or $\arg\max$)P. The surrogate dual is defined as

$$\sup_{u \geq 0} v(SR(u))$$

where $v(SR(u))$ is the value function of the *surrogate relaxation (SR(u)):*

$$\begin{aligned} &\min cx \\ &\text{subject to } u(Ax - b) \geq 0, \quad x \in X. \end{aligned} \tag{SR(u)}$$

Since $v(SR(\theta u)) = v(SR(u))$ for any $\theta \in \mathbb{R}$, the surrogate dual variables are usually normalized, giving the *Surrogate Dual (SD)* as

$$v(SD) := \sup\{v(SR(u)) \mid u \in S^n := \{u \geq 0 \mid eu = 1\}\} \tag{SD}$$

*Email of first author is Natashia.Boland@newcastle.edu.au, of the second author is andy.eb@rmit.edu.au and of the third author is maurostsouk@googlemail.com. This research was supported by the ARC Discovery Grant no. DP0987445.

where $e = (1, 1, \ldots, 1)$.

To date, solution methods for solving the surrogate dual of an integer program have received relatively little attention in the literature, with key early work given in [1, 2, 3, 8] and a more recent study carried out by Sun and Li [4] The algorithms of Sarin et al. [8] and Kim et al. [3] exploit a very useful connection between the surrogate dual and the problem $P(\alpha, u)$ defined by the value:

$$V_\alpha(u) := \max \{u(Ax - b) \mid x \in X(\alpha)\} \qquad (P(\alpha, u))$$

where $X(\alpha) := \{x \in X \mid cx \leq \alpha\}$ and $u \in S^n$ is a fixed value. It is observed that solving $D(\alpha)$, given by

$$v(D(\alpha)) := \min \{V_\alpha(u) \mid u \in S^n\},$$

can provide a bound on the value of the surrogate dual. Specifically for a given α, $v(D(\alpha)) \geq 0$ if and only if $v(SD) \leq \alpha$. Since $D(\alpha)$ is minimization of a piecewise affine convex function, it is amenable to subgradient optimization algorithms; these are the engines used in the methods of [3] and [8], which proceed as follows: solve $D(\alpha)$ (for some initial guess α) and if $v(D(\alpha)) \geq 0$ then the value of α is an upper bound on $v(SD)$ and α is decreased. Otherwise the value of α is a lower bound on $v(SD)$ and α is increased. Once an upper and lower bound have been obtained, an interval search on α may be used to obtain a better estimate. Thus the structure of the algorithms consists of an outer loop on α, searching what we will call the *value space*, and an inner loop (on u) searching the surrogate dual *multiplier space* in order to solve $D(\alpha)$. Fortunately, $D(\alpha)$ does not need to be solved to optimality in all cases. As noted in [8], $\alpha \geq v(SD)$ if and only if for all $u \in S^n$ there exists $x \in X(\alpha)$ such that $u(Ax - b) \geq 0$, so if for some u, $V_\alpha(u) < 0$, no further search in the multiplier space is required; α can immediately be deduced to provide a lower bound on $v(SD)$. In this case, or if $D(\alpha)$ is solved to optimality, we say the status of α is *resolved*, since it is known whether α provides a lower bound or an upper bound on the value of SD. The approach taken to solving $D(\alpha)$ in previous methods has been relatively straightforward. A standard subgradient optimization method, in which at each iteration, a new iterate is found by taking a step along a (projected) subgradient, is used in [3], while in [8] a smoothed (projected) subgradient is used. More recently, Li and Sun [4] suggest a Kelley's cutting plane approach, and also discuss extensions to nonlinear functions.

In this paper, we generalize and extend these algorithms using three key ideas. First, we note that information collected by solution of $P(\alpha, u)$ in the course of solving $D(\alpha)$ for one value of α can be useful in solving $D(\alpha')$ for other values $\alpha' \neq \alpha$, particularly in the case that methods using piecewise affine under-estimators to minimize $v(D(\alpha))$, such as bundle methods, are applied. Hence it may be unprofitable to completely resolve the status of one value of α before attempting the next, and interleaving the search over the value and multiplier spaces (our second idea) could be more efficient on balance. Our third idea is to approach the search over the multiplier space using a bundle trust method. Such methods have been found in the general nonsmooth optimization setting to be highly effective (see e.g. [6]). We adopt a bundle trust approach similar to that successfully used by Linderoth and Wright [5] to solve a stochastic optimization problem. Moreover we are able to efficiently identify descent directions and opt to use these whenever possible so as to make tests for a lower bound on $v(SD)$ based on the sign of $V_\alpha(u)$ as potent as possible. Our numerical experiments confirm that this approach has superior performance to one based on a state of the art subgradient solver applied directly to the minimization of $v(D(\alpha))$.

In what follows, we first, in Section 2, describe our algorithm. In Sections 3 and 4, we provide the theoretical results that underpin the algorithm, ensuring validity of the steps it takes, and proving convergence. We give the results of numerical experiments in Section 5.

2 Searching the Multiplier and Value Spaces

In this section we first describe the general framework of our algorithm. We then give a formal algorithm specification, and discuss details of the algorithm, in particular steps incorporated to safeguard against or deal with potential numerical issues.

Our algorithm maintains an upper bound $\bar{\alpha}$ and a lower bound $\underline{\alpha}$ on the value of the surrogate dual. At each iteration k of the algorithm, a multiplier iterate u_k and a value iterate $\alpha_k \in [\underline{\alpha}, \bar{\alpha}]$ are determined, and a set $X_k = \{x_1, x_2 \ldots, x_{n_k}\}$ of vectors is maintained, where for each $j = 1, \ldots, n_k$, x_j has been obtained by solving $P(\alpha_i, u)$ for some $i \in \{1, \ldots, k\}$, and some u. (We discuss initialization

of the value bounds and iterates later.) Note that X_k always includes a solution of $P(\alpha_j, u_j)$ for all $j = 1, \ldots, k$, and indeed solving $P(\alpha_k, u_k)$ is the first step at iteration k. But if points in $X(\alpha_j)$ are generated in the course of the algorithm by solution of $P(\alpha_j, u')$ for multiplier vectors u' that are not accepted as new iterates, these may also be added to X_k. Now if $V_{\alpha_k}(u_k) < 0$, we can immediately deduce that α_k is a lower bound on the value of the surrogate dual, (its status is resolved), update $\underline{\alpha}$, and go on to iteration $k+1$ with a new value iterate $\alpha_{k+1} > \alpha_k$. Otherwise, i.e. if $V_{\alpha_k}(u_k) \geq 0$, we check to see if x_k is feasible for the MIP, i.e. check if $Ax_k \geq b$. If so, then since $cx_k \leq \alpha_k$, and cx_k is an upper bound on the value of the surrogate dual, we immediately deduce that the status of α_k is resolved, update $\bar{\alpha}$, and go on to iteration $k+1$ with a new value iterate $\alpha_{k+1} < \alpha_k$. In both these cases, the multiplier iterate is unchanged, i.e. $u_{k+1} := u_k$. Other choices could be considered, but since, as we will show later, the minimizers of $D(\alpha_k)$ converge, this seems a reasonable default.

If at iteration k, the status of α_k cannot be resolved in the ways just described, we seek to make progress on solving the problem $D(\alpha_k)$, i.e. on solving

$$\min_{u \in S^n} V_{\alpha_k}(u) := \max \{u\,(Ax - b) \mid x \in X\,(\alpha_k)\}\,. \tag{1}$$

To do this, we observe that the piecewise affine convex function given by

$$\underline{V}_k(u) := \max \{u\,(Ax - b) \mid x \in X_k\,(\alpha_k)\} \tag{2}$$

where $X_k(\cdot)$ is defined by

$$X_k(\alpha) := X \cap \{x \in X_k \mid cx \leq \alpha\},$$

is an underestimate of V_{α_k}, i.e. $\underline{V}_k(u) \leq V_{\alpha_k}(u)$ for all u. Thus the set X_k can be viewed as a kind of "global" set of vectors x_j, each of which induce a subgradient $s_j := Ax_j - b$ of V_α for some α (certainly for $\alpha = \alpha_j$), and the operation $X_k(\alpha_k)$ picks out those that apply "locally" at iteration k, i.e. those for which the half-space

$$\{(u, v) \mid v \geq (Ax_j - b)u\} \supseteq \{(v, u) \mid v \geq V_{\alpha_k}(u)\},$$

i.e. contains the epigraph of $V_{\alpha_k}(\cdot)$. Thus the following definition is convenient.

Definition 1 *If $x \in X(\alpha)$, then $V_\alpha(u) \geq u(Ax - b)$ for all u, and we say $s = Ax - b$ is* valid *for $D(\alpha)$. We will also say it is valid for α, and may also describe the x that induces s as valid for α. Note that if there is some u for which $V_\alpha(u) = u(Ax - b)$ then $v = u(Ax - b)$ is a supporting hyperplane for the epigraph of $V_\alpha(u)$, treated as a function of u, and $s = Ax - b$ is a subgradient of it.*

In this way the set X_k is storing information that is potentially useful for solving $D(\alpha)$ for many values of α, and for a given value of α, those $x \in X_k$ which are most likely to be useful, i.e. those which are valid for α, can readily be picked out by an efficient calculation (the calculation of $X_k(\alpha)$). (Indeed this calculation can be made very efficient by observing that if x is valid for α, then x is also valid for any $\hat{\alpha} \geq \alpha$. One knows x_j is valid for cx_j for each $j = 0, \ldots, k$, so for a new α, placing it in the ordered list of cx_j values immediately divides the indices into those which yield valid points and those which do not.)

The underestimator $\underline{V}_k$ can be treated as a model function for $V_{\alpha_k}(\cdot)$ in a bundle-trust method, where the set $\{Ax - b \mid x \in X_k(\alpha_k)\}$ constitutes the bundle. So progress towards solving $D(\alpha_k)$ can be made by minimizing $\underline{V}_k$ over the current trust region, denoted by B_k, i.e. by solving

$$\begin{aligned} \beta^* := \min_{u,\beta}\ & \beta \\ \text{s.t. } & \beta \geq u\,(Ax - b) \qquad \forall x \in X_k\,(\alpha_k) \\ & u \in S^n \cap B_k, \end{aligned} \tag{3}$$

where B_k could be the whole space or a box around the current multiplier u_k, and acts to control the model fit. Let (β^*, u^*) denote an optimal solution of this problem. There are two cases to consider.

Case 1: If $\beta^* \geq 0$ and gives the minimum value of $\underline{V}_k$, (i.e. $\beta^* = \min_{u \in S^n} \underline{V}_k(u)$, where the minimization is not restricted to the trust region, and we discuss how to check this later), then since $\underline{V}_k$ is an underestimator of V_{α_k}, it must be that $v(D(\alpha_k)) \geq 0$ and hence α_k must be an upper bound on the value of the surrogate dual. Thus the status of α_k is resolved, we update $\bar{\alpha}$, and go on to iteration $k+1$ with a new value iterate $\alpha_{k+1} < \alpha_k$. The new multiplier iterate is set to u^*, i.e. $u_{k+1} := u^*$.

Case 2: In this case u^* is not known to minimize $\underline{V}_k(u)$ over S^n, or its minimum value is negative. Either way the status of α_k cannot immediately be resolved. In this case, any method desired can be used to make progress on solving $D(\alpha_k)$: we require only that it generate a multiplier iterate u_{k+1} that either resolves the status of α_k via the Case 1 tests above, or achieves some measurable descent in V_{α_k}. At this point a new value iterate may also be chosen (or it may be kept the same). We propose to use a bundle-trust method to make progress on solving $D(\alpha_k)$. Specifically, we propose to take "bundle-trust steps" until a new iterate satisfying these conditions has been obtained. In each bundle-trust step, the test for sufficient descent requires solution of $P(\alpha_k, u')$ for some proposed new multiplier vector u'. This generates a vector $x \in X(\alpha_k)$, which we always add to X_k, even if u' is not found to be acceptable as the new iterate u_{k+1}. Note that we don't propose to immediately test such x values for MIP feasibility, but defer that to the next iteration.

The above algorithm framework is quite flexible, and can be viewed as a generalization of previous algorithms. For example, by always choosing the new value iterate in Case 2 to simply be the previous iterate, we obtain methods that fully resolve the status each value of α before trying another value, which has been the traditional approach.

The algorithm we propose is specified formally as follows. In some steps, additional calculations or checks are needed to safeguard against numerical issues; we discuss these afterwards. Note that B_k, the trust region at iteration k, is taken to be $B(\Delta_k)$, where Δ_k is a parameter controlling the size of the trust region. In principle, there are many possible choices of B_k: here we propose to use a box region centered about the current iterate, i.e. define $B_k := B(\Delta_k) = \{u \mid ||u - u_k||_\infty \leq \Delta_k\}$.

Algorithm SDTR

Step 0. set ε, the stopping tolerance parameter, and initialize the trust region size parameter Δ_1
set $\bar{\alpha}$ to some upper bound on $v\,(SD)$ (possibly $+\infty$)
set $\underline{\alpha}$ to some lower bound on $v\,(SD)$ (usually the LP relaxation)
set $k := 1$, initialize $X_1 := \emptyset$ and choose $\alpha_1 \in (\underline{\alpha}, \bar{\alpha})$
initialize u_1, for example to the normalized dual LP solution

Step 1. **if** $|\overline{\alpha} - \underline{\alpha}| \leq \varepsilon$ **then** stop: $v(SD)$ has been found to within the tolerance
else
solve $P\,(\alpha_k, u_k)$ to obtain a solution x_k and calculate the value $V_{\alpha_k}(u_k)$
add x_k to X_k
endif

Step 2. **if** $V_{\alpha_k}(u_k) < 0$ **then**
/ α_k is a lower bound on $v\,(SD)$ */*
update $\underline{\alpha} := \alpha_k$
else if any $x \in X_k$ (not yet checked) is MIP feasible and has $cx < \bar{\alpha}$ **then**
update $\bar{\alpha} := \min\{cx \mid x \in X_k,\ Ax \geq b\}$
else go to Step 3
endif
set $X_{k+1} := X_k$, $\Delta_{k+1} := \Delta_k$, $u_{k+1} := u_k$ and choose $\alpha_{k+1} \in (\underline{\alpha}, \bar{\alpha})$
$k := k + 1$
go to Step 1

Step 3. */* the status of α_k is not yet resolved, so steps toward solving $D(\alpha_k)$ need to be taken */*
solve $\beta^* := \min\{\underline{V}_k(u) \mid u \in S^n \cap B_k(\Delta_k)\}$ by solving (3) to yield solution u^*
/ β^* is the minimum value of the model function over the trust region, and u^* is a minimizer */*
(a) **if** $\beta^* \geq 0$ and u^* solves $\min_{u \in S^n} \underline{V}_k(u)$ **then**
/ u^* minimizes the model function and α_k is an upper bound on $v\,(SD)$ */*
update $\bar{\alpha} := \alpha_k$
set $X_{k+1} := X_k$, $\Delta_{k+1} := \Delta_k$, $u_{k+1} := u^*$ and choose $\alpha_{k+1} \in (\underline{\alpha}, \bar{\alpha})$
$k := k + 1$
go to Step 1
endif
(b) */* take a bundle-trust step toward solving $D(\alpha_k)$ */*
solve problem $P\,(\alpha_k, u^*)$ to obtain minimizer $z \in X(\alpha_k)$ and value $V_{\alpha_k}(u^*)$
/ calculate ρ, the descent in V_{α_k} achieved by u^* relative to that predicted by the model function:*/*

set $\rho := \max\left\{\dfrac{V_{\alpha_k}(u_k) - V_{\alpha_k}(u^*)}{V_{\alpha_k}(u_k) - \beta^*}, 0\right\}$
/ reset the trust region size parameter if needed */*
if $\rho \geq \frac{3}{4}$ and $\|u^* - u_k\|_\infty = \Delta_k$ **then** increase Δ_k by setting $\Delta_k := \bar{\gamma}\Delta_k$ for a fixed $\bar{\gamma} > 1$
else if $\rho \leq \frac{1}{4}$ **then** decrease Δ_k by setting $\Delta_k := \gamma\Delta_k$ for a fixed $\gamma \in (0, 1)$
else */* Δ_k is unchanged */*
endif
/ check to see if demonstrably positive descent has been achieved */*
if $\rho \geq \xi$ (a small positive number, e.g. 10^{-4}) **then**
set $X_{k+1} := X_k \cup \{z\}$, $\Delta_{k+1} := \Delta_k$, $u_{k+1} := u^*$ and choose $\alpha_{k+1} \in (\underline{\alpha}, \bar{\alpha})$
$k := k + 1$
go to Step 1
else */* $\rho < \xi < \frac{1}{4}$ and so Δ_k has just been decreased */*
/ insufficient descent achieved, so no new iterate generated */*
add z to X_k
go to Step 3 (repeat it)
endif

Our theory will show that when all conditions prior to the trust region step fail to be met then we have found a descent direction for $u \mapsto V_{\alpha_k}(u)$ at our current point u_k. When insufficient descent is obtained using this descent direction it likely due to a poor model fit and so we continue to execute trust region steps in a effort to improve the model to get a better descent direction. Effort in aggressively decreasing the value of $V_{\alpha_k}(\cdot)$ results in a more robust test for lower bounding values α_k of $v(SD)$ which is found to be important for improving performance of the algorithm. In Step 3(a) of the algorithm, we need to determine whether or not u^* solves $\min_{u \in S^n} \underline{V}_k(u)$. This can be detected if either (i) $\beta^* = \underline{V}_k(u_k)$, which shows that u_k must lie on a "flat" section of the model function and so must be a minimizer of it, or if (ii) u^* is in the interior of the trust region. This latter test is simply checking whether or not any component of $u^* - u_k$ is in absolute value close to Δ_k. Formal confirmation of these assertions is provided in Lemma 2. In practice, we implement this in an alternative way, which more aggressively seeks to resolve the status of α_k: we (somewhat optimistically) take $\beta^* \geq 0$ to indicate there is some hope that $\min_{u \in S^n} \underline{V}_k(u) \geq 0$, and so go ahead and solve this problem. We may thus minimize the model function twice: once with the trust region and once without. This alternative Step 3(a) is as follows.

(a) **if** $\beta^* \geq 0$ **then**
/ aggressively seek to confirm α_k is an upper bound */*
solve $\min_{u \in S^n} \underline{V}_k(u)$ to obtain solution $\hat{u}$
if $\underline{V}_k(\hat{u}) \geq 0$ **then**
/ $\hat{u}$ minimizes the model function and α_k is an upper bound on $v\,(SD)$ */*
update $\bar{\alpha} := \alpha_k$
set $X_{k+1} := X_k$, $\Delta_{k+1} := \Delta_k$, $u_{k+1} := \hat{u}$ and choose $\alpha_{k+1} \in (\underline{\alpha}, \bar{\alpha})$
$k := k + 1$
go to Step 1
endif
endif

In Step 3(b), we note that from solution of $P\,(\alpha_k, u^*)$ it may be found that $V_{\alpha_k}(u^*) < 0$, in which case α_k is known to be a lower bound. We do not explicitly include this test, since in practice we always truncate solution of $P(\alpha, u)$ for any α and u trialled as soon as it can be deduced that its value is negative (updating the lower bound). This is discussed further in Remark 4 in Section 5.2. In Step 3(b) a number of approaches to selecting α_{k+1} may be considered. As changes in α can result in the shedding of many subgradients in the model function it may well be wise to set α_{k+1} back to a prior value $\alpha_l \in (\underline{\alpha}, \overline{\alpha})$ that has not yet been excluded in the value interval search. Alternatively one may choose to keep α_k constant within this part of the algorithm, i.e. simply set $\alpha_{k+1} := \alpha_k$. Thus the convergence proof must address the possibility that there is an infinite loop of acceptable descents (i.e. an infinite sequence of iterations in which $\rho \geq \xi$) with $[\underline{\alpha}, \overline{\alpha}]$ fixed). An infinite loop of failed descents (or poor model fits) with $\rho < \xi$ in Step 3(b) must also be considered. Our proof in the next sections addresses these issues, and proves convergence for any choice of $\alpha_{k+1} \in (\underline{\alpha}, \bar{\alpha})$. In our implementation

of Step 3(b), we consider two variants for setting α_{k+1} in the case that ρ is strictly positive ($\rho \geq \xi$): we either set $\alpha_{k+1} := \alpha_k$, or we choose $\alpha_{k+1} < \alpha_k$, where at this step $\alpha_k < \bar{\alpha}$. The latter choice reflects our experience that the algorithm performed better if it "focussed" on obtaining lower bounds; seeking to resolve values of α in the lower end of the range was observed to work better in practice.

As the algorithm performs an interval search on the SD value space, it will convergence prematurely if there is any error when setting lower or upper bounds on α. We considered two possibilities for addressing this: (i) the careful and conservative use of tolerances for any tests of sign used to deduce a bound on α, or (ii) validation of the bound via solution of an appropriate surrogate relaxation knapsack problem. We discuss the latter first. As the proposition below (proof of which follows directly from definitions) shows, errors in SD value bound detection can be checked by solving a surrogate relaxation knapsack problem. The proposition notes that solving surrogate relaxations can also strengthen lower bounds, since the optimal value of any surrogate relaxation gives a lower bound on the SD value, possibly better than that currently known. Thus solving surrogate relaxations can have multiple benefits.

Proposition 1 *Suppose at iteration k of the algorithm, either (i) it is calculated in Step 2 that $V_{\alpha_k}(u_k) < 0$ and then calculated that $v(SR(u_k)) < \alpha_k$, or (ii) it is calculated in Step 3(a) that $\min_{u \in S^n} \underline{V}_{\alpha_k}(u) \geq 0$,with minimizer $\hat{u}$ and then calculated that $v(SR(\hat{u})) > \alpha_k$. Then there is a contradiction due to numerical errors. In either case, the value of the surrogate relaxation calculated provides a lower bound on the value of the surrogate dual, and can be used to improve $\underline{\alpha}$, if higher.*

Unfortunately, we found that in practice the surrogate relaxation was often prohibitively slow to solve. We thus abandoned solution of surrogate relaxations, and instead implemented the conservative use of tolerances for tests used to deduce SD bounds. The specifics are provided along with our numerical results in Section 5.

Remark 1 *Many subgradient algorithms that "bundle" subgradients while trying to build a model of the function actually periodically delete "inactive" subgradients from the bundle. If we were to include this in Step 3(b) when u^* is* rejected *as the new multiplier iterate (i.e. when $\rho < \xi$) then we could follow the strategy of [5]. This strategy possibly deletes subgradients when they have not been active in the solution of (3) for many iterates or fail a comparative descent test (i.e. equation (20) of [5]). This is because we need to establish a lemma similar to Lemma 4 of [5]. Currently we keep all subgradients generated during this loop in which α_k remains fixed.*

3 On the Detection of Descent Directions

Throughout this paper in proofs we will need to assume $V_\alpha(\cdot)$ is finite valued. A convenient (but not essential assumption) to ensure this is to assume that $X(\alpha)$ is a bounded set. As usual we seek directions d that are descent directions for $u' \mapsto V_\alpha(u')$ at $u' = u$, that is, $V_\alpha(u + \gamma d) < V_\alpha(u)$ for all sufficiently small $\gamma > 0$. In an unregulated descent from the current value u_j using the subgradient $Ax_j - b$, the new iterate u_{j+1} is found by solution of

$$\min\left\{V_{\alpha_j}(u_j) + (u - u_j)(Ax_j - b) \mid u \in S^n\right\} = \min\left\{u(Ax_j - b) \mid u \in S^n\right\},$$

which gives is the maximum descent predicted by the linear approximation of $u \mapsto V_{\alpha_j}(u)$ around u_j. If $\hat{u}$ is the solution found, then $u_{j+1} := u_j + (\hat{u} - u_j) = \hat{u}$. Subgradient solvers of the 1970's simply modified this by using a predefined step size λ^k to ensure convergence and selected $u_{j+1} := u_j + \lambda^k(\hat{u} - u_j)$. When $(\hat{u} - u_j)(Ax_j - b) < 0$ we have a potential descent direction $d_{j+1} := (\hat{u} - u_j)$ for $u \mapsto V_{\alpha_j}(u)$ as is shown in the next lemma. It is well known that taking a full step may result in violent variations in u (which may not provide actual descent for $u \mapsto V_{\alpha_j}(u)$ in each case). Thus a smaller step along d_{j+1} might be advisable. In [2] it is noted that sufficient conditions for an ascent direction of the surrogate relaxation cannot be provided. This is not the case for the dual function $u \mapsto V_\alpha(u)$ as was first noticed in [2] in Theorem 5.1. We improve on this result via the addition of the observations contained in Lemma 1 part 3.

Remark 2 *Observe that $V_\alpha(u)$ can be interpreted as a Lagrangian relaxation of the problem*

$$\max\{0 : Ax \geq b, x \in X(\alpha)\}.$$

The characterization of the subdifferential of V_α thus follows from well-known results from Lagrangian relaxation and optimization of functions of the form $z_{LR}(u) = \max_{k=1,\ldots,K} u^T(Ax^k - b)$, where here $\{x^1, \ldots, x^K\}$ is the finite set of extreme points of $\mathrm{conv}(X(\alpha))$. Then clearly $V_\alpha(0) = 0$ and for all $s \in \partial V_\alpha(u)$ we have $V_\alpha(u) = su$ and $V_\alpha(w) \geq sw$ for all w. So

$$\partial V_\alpha(u) = \{s \mid s = Ax - b \text{ for all } x \in X(\alpha) \text{ with } V_\alpha(u) = su\}. \tag{4}$$

Lemma 1 *Suppose $X(\alpha)$ is a bounded set (so $V_\alpha(u) < +\infty$). The following are equivalent.*

1. *d is a descent direction for $u' \mapsto V_\alpha(u')$ at $u' = u$.*
2. *$d(Ax - b) < 0$ for all $x \in \arg\max\{u(Ax' - b) \mid x' \in X(\alpha)\}$.*
3. *$ds < 0$ for all*

$$s := (Ax - b) \in \partial_d V_\alpha(u) := \{s \in \partial V_\alpha(u) \mid \langle s, d\rangle = V'_\alpha(u, d)\}, \tag{5}$$

where

$$V'_\alpha(u, d) = \lim_{t \downarrow 0} \frac{V_\alpha(u + td) - V_\alpha(u)}{t}.$$

In particular when there exists $x_\gamma \in \arg\max\{(u + \gamma d_\gamma)(Ax' - b) \mid x' \in X(\alpha)\}$ as $\gamma \downarrow 0$ with $d_\gamma \to d$, $x_\gamma \to x$ and $d(Ax - b) < 0$ then d is a descent direction.

Proof. Suppose $V_\alpha(u+\gamma d) < V_\alpha(u) < +\infty$ for some small $\gamma > 0$ and $x \in \arg\max\{u(Ax' - b) \mid x' \in X(\alpha)\}$ then $x \in X(\alpha)$ and so

$$\begin{aligned}\gamma d(Ax - b) &\leq \max\{(u + \gamma d)(Ax' - b) \mid x' \in X(\alpha)\} - u(Ax - b)\\ &= V_\alpha(u + \gamma d) - V_\alpha(u) < 0\end{aligned}$$

implying $d(Ax - b) < 0$. As this is true for all $x \in \arg\max\{u(Ax' - b) \mid x' \in X(\alpha)\}$ it is also true for all s as in (5).

Conversely for all

$$x_\gamma \in M(\gamma) := \arg\max\{(u + \gamma d)(Ax' - b) \mid x' \in X(\alpha)\}$$

we have

$$\begin{aligned}V_\alpha(u + \gamma d) &= (u + \gamma d)(Ax_\gamma - b)\\ &= u(Ax_\gamma - b) + \gamma d(Ax_\gamma - b) \leq V_\alpha(u) + \gamma d(Ax_\gamma - b).\end{aligned}$$

Next we note that as $M(0)$ is bounded and $g_\gamma(x) := (u + \gamma d)(Ax - b) + \delta_{X(\alpha)}(x)$ is a sequence of level set bounded, convex functions epi-converging to $g(x) := u(Ax - b) + \delta_{X(\alpha)}(x)$ we may invoke [7], Exercise 7.32 (c) and Theorem 7.33 to deduce $\limsup_{\gamma \downarrow 0} M(\gamma) \subseteq M(0) = \arg\max\{u(Ax' - b) \mid x' \in X(\alpha)\}$. When we assume $d(Ax - b) < 0$ for all $x \in M(0)$ we have for γ sufficiently small $d(Ax_\gamma - b) < 0$. If we assume otherwise (i.e. there exists $x_{\gamma_m} \in M(\gamma_m)$ for $\gamma_m \downarrow 0$ with $d(Ax_{\gamma_m} - b) \geq 0$) the following contradictions follows. As $V_\alpha(\cdot)$ is a finite convex function it is Locally Lipschitz and so $\partial V_\alpha(\cdot)$ is locally uniformly bounded imply local boundedness of $s_\gamma := (Ax_\gamma - b)$. This in turn implies local boundedness of $\{x_{\gamma_m}\}$ and on taking any convergent subsequence $x_{\gamma_{m_k}} \to x \in M(0)$ we find that the assumption that $d\left(Ax_{\gamma_{m_k}} - b\right) \geq 0$ implies the contradiction $d(Ax - b) \geq 0$ for some $x \in M(0)$. Thus $d(Ax_\gamma - b) < 0$ for γ small and

$$V_\alpha(u + \gamma d) \leq V_\alpha(u) + \gamma d(Ax_\gamma - b) < V_\alpha(u).$$

If there exists a convergent sequence $x_{\gamma_m} \in M(\gamma_m)$ for $\gamma_m \downarrow 0$ such that $x_{\gamma_m} \to x \in M(0)$ with $d(Ax - b) < 0$ then the same argument implies d is a descent direction. Indeed the presumption that there exists a further subsequences such that $d\left(Ax_{\gamma_{m_k}} - b\right) \geq 0$ for all k implies the contradiction $d(Ax - b) \geq 0$. Thus we have $\gamma_m d(Ax_{\gamma_m} - b) < 0$ for m large implying $V_\alpha(u + \gamma_m d) < V_\alpha(u)$.

For the third equivalence we observe that when $x_\gamma \in M(\gamma)$ we have

$$s_\gamma := (Ax_\gamma - b) \in \partial V_\alpha(u + \gamma d)$$

and as $V_\alpha(\cdot)$ is a finite convex function it is also semi-smooth. Consequently any convergent subsequence $s_{\gamma_m} := (Ax_{\gamma_m} - b) \to s = (Ax - b) \in \partial_d V_\alpha(u)$ for some $x \in M(0)$. On assumption that $d(Ax - b) < 0$ for $s = (Ax - b) \in \partial_d V_\alpha(u)$ we may again argue as above that for γ sufficiently small $d(Ax_\gamma - b) < 0$ and hence $V_\alpha(u + \gamma d) < V_\alpha(u)$. ■

A model function may be built up by collecting a set of subgradients. One may then perform any one of a number of methods to find an approximate minimum of the model function (possibly subject to a trust region). This model function would be a piecewise affine lower approximation $\underline{V}_k$ of $u \mapsto V_{\alpha_k}(u)$ as described in (2). The algorithm sets up a series of conditional tests that are designed to extract as much information about current value α_k in relation to $v(SD)$ before resorting to descent of V_{α_k} at u_k using the model function $\underline{V}_k$ to find a descent direction.

In the following analysis we assume that we are at a point $u_k \in S^n$ with $V_{\alpha_k}(u_k) \geq 0$ (since $V_{\alpha_k}(u_k) < 0$ implies α_k is a lower bound of $v(SD)$). Suppose u_{k+1} solves this problem and j is an active constraint. As $Ax_k - b$ is a subgradient of $V_{\alpha_k}(\cdot)$ at u_k and $x_k \in X_k(\alpha_k)$ we have

$$\begin{aligned} u_k(Ax_k - b) = V_{\alpha_k}(u_k) &\geq \underline{V}_k(u_k) \\ &\geq \min_{u \in S^n \cap B} V_{\alpha_k}(u) = \beta_{k+1} = u_{k+1}(Ax_j - b) \geq u_{k+1}(Ax_k - b) \end{aligned}$$

and so

$$0 \geq \left(u^{k+1} - u^k\right)(Ax_k - b).$$

Now if we actually get a descent on the model function then

$$\underline{V}_k(u_k) > \min_{u \in S^n \cap B} \underline{V}_k(u) = \beta_{k+1} \tag{6}$$

and hence we may strengthen the inequality to

$$0 > (u_{k+1} - u_k)(Ax_k - b) \quad \text{for the given } x_k \in \arg\max\{u_k(Ax - b) \mid x \in X(\alpha_k)\}. \tag{7}$$

When $\arg\max\{u_k(Ax - b) \mid x \in X(\alpha_k)\}$ is unique Lemma 1 implies $d_{k+1} := (u_{k+1} - u_k)$ is a descent direction for the actual function $u' \mapsto V_{\alpha_k}(u')$ at $u' = u_k$. Again this does not mean we can take a full step to u_{k+1} and obtain a descent in $V_{\alpha_k}(\cdot)$ and if $\arg\max\{u_k(Ax - b) \mid x \in X(\alpha_k)\}$ is not unique then an actual descent might not exist. In this case we may need to improve the model function in order to detect this. In short we must check to see if d_{k+1} results in an actual descent of the real function $V_{\alpha_k}(\cdot)$. Thus we propose to use a trust region strategy to vary the step length and model function so as to ensure real descent of the underlying function V_α. Two options are explored in this paper, the use of an external subgradient bundle solver and a trust region method based on the CPLEX linear programming solver as the engine.

When we find the alternative $\beta_{k+1} < 0$ then as $V_{\alpha_k}(u_k) \geq 0$ it follows that

$$u_k(Ax_k - b) = V_{\alpha_k}(u_k) > \min_{u \in S^n \cap B} \underline{V}_k(u) = \beta_{k+1} \geq u_{k+1}(Ax_k - b)$$

and (7) ensues. As $\beta_{k+1} < 0$ can occur due a poor model function we again must test for actual descent via a trust region test.

When (6) fails we have then verified that u_k is a local minimum of $\underline{V}_k(\cdot)$ relative to S^n. This is easily see when u_k is *not* a boundary point of $B(\Delta)$ (we usually choose $B(\Delta_k) := \{u \mid \|u - u_k\|_\infty \leq \Delta_k\}$). In this case we have u_k a local and hence also a global minimum of $\underline{V}_k(\cdot)$ relative to S^n, due to convexity. When u_k is a boundary point we require further analysis to establish this fact.

Lemma 2 *Suppose $V_{\alpha_k}(u_k) \geq 0$ and $B(\Delta_k) := \{u \mid \|u - u_k\|_\infty \leq \Delta_k\}$. If (6) fails i.e.*

$$\underline{V}_k(u_k) = \min_{u \in S^n \cap B} \underline{V}_k(u) = \beta_{k+1}$$

then u_k is a global minimum of $u \mapsto \underline{V}_k(u)$. Then $\beta_{k+1} \geq 0$ implies $\alpha_k \geq v(SD)$. If (6) fails and $V_{\alpha_k}(u_k) = 0$ we have $v(D(\alpha_k)) = 0$. Moreover when $u_{k+1} \in \operatorname{int} B(\Delta_k)$ it also follows that u_{k+1} is a global minimum of $u \mapsto \underline{V}_k(u)$.

Proof. Let $u \in S^n$ and let $\lambda \in (0, 1)$ be such that $\bar{u} := \lambda u + (1 - \lambda) u_k \in B$ (i.e. we draw a line from u to u_k and note it's intersection with the boundary of B). Consequently

$$\begin{aligned} 0 \leq \underline{V}_k(\bar{u}) - \underline{V}_k(u_k) &= \underline{V}_k(\lambda u + (1 - \lambda) u_k) - \underline{V}_k(u_k) \\ &\leq \lambda \underline{V}_k(u) + (1 - \lambda) \underline{V}_k(u_k) - \underline{V}_k(u_k) \end{aligned}$$

and so

$$\underline{V}_k(u_k) \leq \underline{V}_k(u).$$

When $u_{k+1} \in \operatorname{int} B$ we can replace u_k by u_{k+1} in the above argument and deduce that u_{k+1} is a global minimum of $\underline{V}_k(\cdot)$.

The model function $\underline{V}_k(\cdot)$ is always a lower minorant of $V_{\alpha_k}(\cdot)$. Thus the minimum value β_{k+1} of $\underline{V}_k(\cdot)$ is also a minorant of the minimum value of $V_{\alpha_k}(\cdot)$ so when $\beta_{k+1} \geq 0$ we have

$$v(D(\alpha_k)) \geq \beta_{k+1} \geq 0$$

implying an upper bound $v(SD) \leq \alpha_k$. When (6) fails and $V_{\alpha_k}(u_k) = 0$ we have

$$0 = V_{\alpha_k}(u_k) = v(D(\alpha_k)) \geq \beta_{k+1} \geq 0$$

implying $v(D(\alpha_k)) = 0$. ■

Thus when we have not terminated on the condition $\beta_{k+1} \geq 0$ we find that solving (3) gives (7) and a potential descent direction.

We still need to determine the step length to actually obtain a descent in the underlying convex function $V_{\alpha_k}(\cdot)$ so one can utilize a trust region strategy to obtain this. This differs from most other trust region algorithms in that our model function is actually polyhedral not quadratic. Similar consideration arises in [5] which provides some valuable tools for convergence analysis. The SDTR algorithm differs from other cutting plane like algorithms for the surrogate dual proposed in [4] and [2], say, in that a trust region strategy is employed to regulate (and ensure) a descent and in that we do not need to solve the surrogate relaxation at any stage. The surrogate relaxation may be employed (optionally) to check the validity of bounds but our numerical experience has shown that the computational effort required to solve the surrogate relaxation is unpredictable while the computational effort for the problem $P(\alpha, u)$ is always low. This we opt to avoid the solution of the surrogate relaxation. Thus one can view this as an entirely dual based algorithm that attempts to exhaustively search the multiplier space.

4 Convergence Analysis for Algorithm SDTR

As the basic surrogate dual algorithm is based on an interval search it can only fail if this interval bisection is not returned to infinitely often. Thus methods based on the global minimization of $u \mapsto V_\alpha(u)$ in step 3(b) must converge without further analysis. Thus the convergence analysis pivots only on the trust region method that has no a-priori exit route from inner iterates that tries to descend the function $u \mapsto V_\alpha(u)$. To establish that the algorithm SDTR will ultimately be successful we need to first prove that the trust region loop inside step 3 of SDTR will not indefinitely reject u_{k+1} and fail to achieve sufficient decent. We also need to show that there is no infinite loop involving acceptance of a descent direction but a reduction of Δ to zero due to a string of bad model fits. The first Lemmas and its corollaries are essential in establishing these cases do not occur. The first Lemma deals with the case of a bad model fit and an associated $\Delta_k \downarrow 0$ and the second with an unbroken sequence of insufficient descent.

Denote the support functions to a closed convex set A by $\delta^*(A)(u) := \sup\{au \mid a \in A\}$. We use epi-limits and Attouch's theorem for which the reader may consult [7] for details. We note that in step 3 of SDTR, if the iterate fails the test $\rho_{k+1} > \xi$ then $\alpha_{k+1} = \alpha_k$ and if it passes the test (for a MIP) $\alpha_{k+1} := cx_k$. In both cases we have $x_{k+1} \in X_{k+1}(\alpha_{k+1})$. We say $u_k \to_d \bar{u}$ if $\frac{u_{k+1}-u_k}{\|u_{k+1}-u_k\|_\infty} \to d$ as $k \to \infty$.

Lemma 3 *Assume $\alpha_k \downarrow \alpha$ with $X(\alpha_k)$ eventually bounded (so $V_{\alpha_k}(\cdot) < +\infty$). Suppose*

$$\{B_k := \{v \mid \|v - u_k\|_\infty \leq \Delta_k\}\}_{k=0}^\infty$$

is an arbitrary sequence of trust regions for any infinite (unbroken) sequence of trust region loops through step 3, along which $x_{k+1} \in X_{k+1}(\alpha_{k+1})$ with $\Delta_k \downarrow 0$. Suppose in addition that there exists a subsequence $\{d_{k_p}\}_{p=0}^\infty$ of $\left\{d_k := \frac{u_{k+1}-u_k}{\|u_{k+1}-u_k\|_\infty}\right\}_{k=0}^\infty$ such that $d_{k_p} \to d$ with $u_{k_p} \to u$ and $V'_\alpha(u, d) < 0$. Then

$$\limsup_{k\to\infty} \rho_{k+1} = 1.$$

Proof. Along with a sequence of trust regions of diameter $\Delta_k \downarrow 0$ we have an associated sequence of u_{k+1} generated in the evaluation of $V_{\alpha_k}(u_{k+1})$ when calculating ρ_{k+1}. In step 3 or SDTR we add some

$$s_{k+1} \in \partial V_{\alpha_k}(u_{k+1})$$

by choosing $s_{k+1} = Az_{k+1} - b$.

By assumption $\|u_{k+1} - u_k\|_\infty \to 0$. As $u_k \to u$ we have $\|u_{k+1} - u\|_\infty \to 0$. By assumption there exists a convergent subsequence of $\left\{d_k := \frac{u_{k+1}-u_k}{\|u_{k+1}-u_k\|_\infty}\right\}_{k=0}^{\infty}$ and denoting this sequence by $\{d_p\}_{p=0}^{\infty}$ we have $\lim_{p\to\infty} d_p = d$ such that $v'(P(\alpha,\cdot))(u,d) < 0$. Then denoting $t_p := \left\|u_{k_p+1} - u_{k_p}\right\|_\infty$ we have

$$u_{k_p+1} - u_{k_p} = t_p d_p.$$

Note that in step 3 (b) just before returning to to do step 3 again, we add

$$z_{k_p+1} \in \arg\max\left\{u_{k_p+1}(Ax-b) \mid x \in X\left(\alpha_{k_p}\right)\right\}$$

found while calculating ρ_{k_p+1}. This implies $\left(Az_{k_p+1} - b\right)$ must satisfy

$$\begin{aligned} V_{\alpha_{k_p}}(u_{k_p} + t_p d_p) &= \left(Az_{k_p+1} - b\right) \cdot \left(u_{k_p} + t_p d_p\right) \\ &\geq s \cdot \left(u_{k_p} + t_p d_p\right), \quad \forall s = (Ax-b),\, x \in X_{k_p}\left(\alpha_{k_p}\right) \end{aligned} \tag{8}$$

at each iteration.

Let $y_p \in X\left(\alpha_{k_p}\right)$ satisfy $\beta_{k_p+1} = u_{k_p+1}(Ay_p - b)$ in the problem (3) from which we obtain $\left(u_{k_p+1}, \beta_{k_p+1}\right)$. Due to (8) we have

$$V_{\alpha_{k_p}}(u_{k_p} + t_p d_p) \geq (Ay_p - b) \cdot \left(u_{k_p} + t_p d_p\right) = \beta_{k_p+1} \tag{9}$$

as $y_p \in X_{k_p}\left(\alpha_{k_p}\right)$. Hence when (7) holds we have

$$0 > V_{\alpha_{k_p}}(u_{k_p+1}) - V_{\alpha_{k_p}}(u_{k_p}) \geq \beta_{k_p+1} - V_{\alpha_{k_p}}(u_{k_p})$$

and so $\rho_{k_p+1} \leq 1$. Note that (7) holds in either of the cases $\beta_{k+1} < 0$ or $\beta_{k+1} < \underline{V}_k(u_k)$.

Now we wish to invoke Attouch's theorem [7], Theorem 12.35 that states that an epi-convergent family of convex functions have their subdifferentials graphically converging. First we note that as $\alpha_{k_p} \downarrow \alpha$ and $X\left(\alpha_{k_p}\right)$ are eventually bounded we have a monotonically non-increasing, pointwise convergent and proper family of convex functions $\left\{V_{\alpha_{k_p}}(\cdot)\right\}_{p=0}^{\infty}$. Consequently this family also epi-converges to $V_\alpha(\cdot)$ and so $\left\{\partial V_{\alpha_{k_p}}(\cdot)\right\}_{p=0}^{\infty}$ graphically converges to $\partial V_\alpha(\cdot)$.

We need to estimate the magnitude of

$$\begin{aligned} \rho_{k_p+1} &\geq \frac{V_{\alpha_{k_p}}\left(u_{k_p+1}\right) - V_{\alpha_{k_p}}\left(u_{k_p}\right)}{\beta_{k_p} - V_{\alpha_{k_p}}\left(u_{k_p}\right)} \\ &= \left(\frac{V_{\alpha_{k_p}}\left(u_{k_p} + t_p d_p\right) - V_{\alpha_{k_{m_p}}}\left(u_{k_{m_p}}\right)}{t_p}\right)\left(\frac{\beta_{k_p} - V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p}\right)^{-1} \end{aligned} \tag{10}$$

As z_{k_p} was added to the model in step step 3(b) at iteration $k_p - 1$ and by assumption is not dropped we have $z_{k_p} \in X_{k_p}\left(\alpha_{k_p}\right)$ when solving (3) at iteration k_p. Thus we have at the iteration k_p a subgradient $s_{k_p} := Az_{k_p} - b$ satisfying $V_{\alpha_{k_p-1}}\left(u_{k_p}\right) = u_{k_p}\left(Az_{k_p} - b\right)$. As $z_{k_p} \in X_{k_p}\left(\alpha_{k_p}\right)$ it follows that $\beta_{k_p+1} \geq u_{k_p+1}s_{k_p} = s_{k_p}\left(u_{k_p} + t_p d_{l_p}\right)$. As $s_{k_p} \in \partial V_{\alpha_{k_p-1}}\left(u_{k_p}\right)$, taking a further subsequence and re-numbering we may assume we have such a sequence of subgradients for which $s_{k_p} \to s = (Ax-b) \in \partial V_\alpha(\bar{u})$ for some $x \in X(\alpha)$ (local Lipshitzness of $u \mapsto V_\alpha(u)$ ensures $\partial V_\alpha(\bar{u})$ is bounded and by the graphical convergence of subdifferentials and the sequence $\{s_p\}$ is locally bounded, see [7] Exercise 5.34(b)).

As (7) applies to each problem in this sequence we have $\frac{V_{\alpha_{k_p}}\left(u_{k_p}+t_p d_p\right)-V_{alpha_{k_p}}\left(u_{k_p}\right)}{t_p}<0$. Now use (9) and the monotonic noncreasingness of $\left\{V_{\alpha_{k_p}}(\cdot)\right\}_{p=0}^{\infty}$ to obtain

$$\begin{aligned}0>&\frac{V_{\alpha_{k_p}}\left(u_{k_p}+t_p d_p\right)-V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p}\geq\frac{\beta_{k_p+1}-V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p}\\ \geq&\frac{s_{k_p}\left(u_{k_p}+t_p d_p\right)-V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p}\geq\frac{s_{k_p}\left(u_{k_p}+t_p d_p\right)-V_{\alpha_{k_p-1}}\left(u_{k_p}\right)}{t_p}=s_{k_p}d_p\end{aligned}$$

where the last equality holds due to the fact that $s_{k_p}\in\partial V_{\alpha_{k_p-1}}\left(u_{k_p}\right)$ implies $V_{\alpha_{k_p-1}}\left(u_{k_p}\right)=s_{k_p}u_{k_p}$. Due to Attouch's theorem $s_{k_p}\to s\in\partial V_\alpha(u)$ so $s_{k_p}d_p\to sd\leq V'_\alpha(u,d)$.

As $\left\{\partial V_{\alpha_{k_p}}(\cdot)\right\}_{p=0}^{\infty}$ graphically converges and this implies the epi-convergence of the associated support functions (see [7], Corollary 11.36) we have for all $w_p\to u$

$$\begin{aligned}\limsup_{\substack{w_p\to u\\ p\to\infty}}\inf_{d'\to d}V'_{\alpha_{k_p}}(w_p,d')&=\lim_{\substack{w_p\to u\\ p\to\infty}}\inf_{d'\to d}\delta^*\left(\partial V_{\alpha_{k_p}}(w_p)\right)(d')\\ &=\delta^*\left(\partial V_\alpha(u)\right)(d)=V'_\alpha(u,d).\end{aligned}$$

As $\partial V_{\alpha_{k_p}}(w_p)$ is locally uniformly bounded we have $d'\mapsto\delta^*\left(\partial V_{\alpha_{k_p}}(w_p)\right)(d')$ uniformly locally Lipschitz and hence

$$\begin{aligned}\limsup_{\substack{w_p\to u\\ p\to\infty}}\inf_{d'\to d}V'_{\alpha_{k_p}}(w_p,d')&=\limsup_{\substack{w_p\to u\\ p\to\infty}}\delta^*\left(\partial V_{\alpha_{k_p}}(w_p)\right)(d)\\ &=\lim_{\substack{w_p\to u\\ p\to\infty}}V'_{\alpha_{k_p}}(w_p,d)=V'_\alpha(u,d)<0.\end{aligned}\tag{11}$$

Consequently for p large we have $V'_{\alpha_{k_p}}(w_p,d)<0$ for any $w_p\to u$.

Using the mean value theorem for convex functions we have the existence of $\gamma_p\in(0,1)$ and $v_p\in\partial V_{\alpha_{k_p}}\left(u_{k_p}+t_p\gamma_p d_p\right)$ such that

$$V'_{\alpha_{k_p}}\left(u_{k_p}+t_p\gamma_p d,d\right)\geq v_p d_p=\frac{V_{\alpha_{k_p}}\left(u_{k_p}+t_p d\right)-V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p}$$

and for p large $V'_{\alpha_{k_p}}\left(u_{k_p}+t_p\gamma_p d,d\right)<0$. Using (10) and observing both quantities in the quotient for ρ_{k_p+1} are negative gives

$$\begin{aligned}\limsup_{p\to\infty}\rho_{k_p+1}&\geq\left(\limsup_{\substack{t_p\downarrow 0,d_p\to d\\ u_{k_p}\to u}}\frac{V_{\alpha_{k_p}}\left(u_{k_p}+t_p d_p\right)-V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p}\right)\left(s_{k_p}d_p\right)^{-1}\\ &=\left(\limsup_{\substack{t_p\downarrow 0\\ u_{k_p}\to u}}\frac{V_{\alpha_{k_p}}\left(u_{k_p}+t_p d\right)-V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p}\right)\left(s_{k_p}d_p\right)^{-1}\quad\text{(Lipschitzness of } V_{\alpha_{k_p}}(\cdot)\text{)}\\ &\geq\left[\limsup_{p\to\infty}V'_{\alpha_{k_p}}\left(u_{k_p}+t_p\gamma_p d,d\right)\right](sd)^{-1}\geq\left[\limsup_{\substack{w_p\to u\\ p\to\infty}}V'_{\alpha_{k_p}}(w_p,d)\right](sd)^{-1}.\end{aligned}$$

Thus

$$\limsup_{k\to\infty}\rho_{k+1}\geq V'_\alpha(u,d)\times(sd)^{-1}\geq 1.$$

■

There is an inner loop of trust region iteration that can fail to find an acceptable descent and hence α remains constant fixed at α_k and we do not update u_k either. Thus it is best to count these iterates and associated variable via a second index i.e. $u_{k,l}$ for $l=1,\ldots,l_k$ where l_k is the final iterate where $\rho_{k,l_k}\geq\xi$ and hence $u_{k+1}=u_{k,l_k}$. The case when when do not vary α_k require less assumptions.

Lemma 4 *Assume $X(\alpha_k)$ is bounded and we take $\Delta_{k+1} = \gamma\Delta_k$ with $\gamma \in (0,1)$ in step 3(b). Suppose $\{\Delta_{k,l_m}\}_{m=0}^{\infty}$ is a arbitrary sequence of trust region of diameters with $\Delta_{k,l_m} \downarrow 0$ in any trust region loop in step 3 with $\alpha = \alpha_k$ and $u = u_k$ fixed (i.e. we keep getting $\rho_{k+1} < \xi$). Then we must have a subsequence $\{\rho_{k,l_p}\}_{p=0}^{\infty} \subseteq \{\rho_{k,l_m}\}_{m=0}^{\infty}$ with*

$$\limsup_{p\to\infty} \rho_{k,l_p} \geq 1.$$

Proof. The argument above in Lemma 3 may be applied with α_k fixed. Take a subsequence $\{d_{l_p}\}_{p=0}^{\infty}$ of $\{d_l := \frac{u_{k,l}-u_k}{|u_{k,l}-u_k|}\}_{p=0}^{\infty}$ converging to d. Denote the subsequence of diameter $\Delta_{k,l_p} \downarrow 0$ as $p \to +\infty$ associated with a sequence of u_{k,l_p} with $u_{k,l_p} \to u_k$. Place $u_{k,l_p} = u_k + t_p d_{l_p}$ and note that we continue to reject an update of u_k. As $u \mapsto V_{\alpha_k}(u)$ is convex it is also semi-smooth and so we may also assume $s_{l_p} \to s$ along the direction d giving

$$s \in \partial_d V_{\alpha_k}(u_k) := \left\{ s' \in \partial V_{\alpha_k}(u_k) \mid s' \cdot d = V'_{\alpha_k}(u_k, d) \right\}.$$

As z_{k,l_p} is added to $X_{k,l_p}(\alpha_k)$ and retained at each iteration we have $z_{k,l_{p-1}} \in X_{k,l_{p-1}}(\alpha_k)$ when solving (3) at iteration (k, l_p) (and α_k is never updated). Consequently $\beta_{k,l_p} \geq u_{k,l_p} \cdot s_{l_p-1} = s_{l_p-1} \cdot (u_k + t_p d_p)$. As (7) applies to each problem in this sequence we have $\frac{V_{\alpha_k}(u_k+t_p d_p) - V_{\alpha_k}(u_k)}{t_p} < 0$. Now use (9) with $\alpha_{k_p} = \alpha_k$ to obtain

$$\begin{aligned}
0 &> \frac{V_{\alpha_k}\left(u_k + t_p d_{l_p}\right) - V_{\alpha_k}(u_k)}{t_p} \\
&\geq \frac{\beta_{k,l_p} - V_{\alpha_k}(u_k)}{t_p} \geq \frac{s_{l_p-1} \cdot (u_k + t_p d_p) - V_{\alpha_k}(u_k)}{t_p} \\
&= s_{l_p-1} \cdot \frac{\left(t_p d_{l_p} - t_{p-1} d_{l_{p-1}}\right)}{t_p} + \frac{s_{l_p-1} \cdot \left(u_k + t_{p-1} d_{l_{p-1}}\right) - V_{\alpha_k}(u_k)}{t_p} \\
&= \left[\left(s_{l_p-1} \cdot d_{l_p}\right) - \frac{t_{p-1}}{t_p}\left(s_{l_p-1} \cdot d_{p-1}\right)\right] + \left(\frac{t_{p-1}}{t_p}\right)\left(\frac{V_{\alpha_k}\left(u_k + t_{p-1} d_{l_{p-1}}\right) - V_{\alpha_k}(u_k)}{t_{p-1}}\right)
\end{aligned}$$

as $s_{l_p-1} \in \partial V_{\alpha_k}(u_k + t_{p-1} d_{p-1})$ implies $V_{\alpha_k}(u_k + t_{p-1} d_{p-1}) = s_{l_p-1} \cdot (u_k + t_{p-1} d_{p-1})$. Suppose first that there exists a subsequence so that $\left\{\frac{t_{p_m-1}}{t_{p_m}}\right\}$ converges to $\lambda \geq 0$. Hence $s_{l_p-1} \cdot d_p \to s \cdot d = V'_{\alpha_k}(u_k, d)$ and $s_{p-1} \cdot d_{p-1} \to V'_{\alpha_k}(u_k, d)$ giving

$$\begin{aligned}
&\left[\left(s_{p_m-1} \cdot d_{p_m}\right) - \frac{t_{p_m-1}}{t_{p_m}}\left(s_{p_m-1} \cdot d_{p_{m-1}}\right)\right] + \left(\frac{t_{p_m-1}}{t_{p_m}}\right)\left(\frac{V_{\alpha_k}\left(u_k + t_{p_m-1} d_{p_{m-1}}\right) - V_{\alpha_k}(u_k)}{t_{p_m-1}}\right) \\
&\to_{m\to\infty} [1-\lambda] V'_{\alpha_k}(u_k, d) + \lambda V'_{\alpha_k}(u_k, d) = V'_{\alpha_k}(u_k, d).
\end{aligned}$$

Hence using (10) and noting both quantities in the quotient are negative gives

$$\begin{aligned}
\limsup_{p\to\infty} \rho_{k+l_p} &\geq \limsup_{p\to\infty} \left(\frac{V_{\alpha_k}\left(u_k + t_p d_{l_p}\right) - V_{\alpha_k}(u_k)}{t_p}\right)\left(\frac{\beta_{k,l_p} - V_{\alpha_k}(u_k)}{t_p}\right)^{-1} \\
&\geq \left(\lim_{t_p \downarrow 0, d_{l_p} \to d} \frac{V_{\alpha_k}\left(u_k + t_p d_{l_p}\right) - V_{\alpha_k}(u_k)}{t_p}\right)\left(V'_{\alpha_k}(u_k, d)\right)^{-1} = 1.
\end{aligned}$$

In the alternative case $\left\{\frac{t_{p-1}}{t_p}\right\}$ is unbounded (and $\Delta_{k,l_p+1} = \gamma\Delta_{k,l_p}$ with $\gamma \in (0,1)$) then $u_k + t_{p-1} d_{p-1} \in \operatorname{int} B_{k,l_p}$ placing us in step 3(a), as described in Lemma 2, resulting in an update of u_k contrary to assumption. ■

Putting both Lemmas 3 and 4 together we obtain the following result.

Corollary 1 *Assume $\alpha_k \downarrow \alpha$ with $X(\alpha_k)$ eventually bounded (so $V_{\alpha_k}(\cdot) < +\infty$). Suppose*

$$\{B_k := \{u \mid \|u - u_k\|_\infty \leq \Delta_k\}\}_{k=0}^{\infty}$$

is an arbitrary sequence of trust regions for any infinite (unbroken) sequence of trust region loops through 3(b) along which $x_{k+1} \in X_{k+1}(\alpha_{k+1})$ with $\Delta_k \downarrow 0$ and $u_k \to u$ for which either:

1. *we have* $\alpha = \alpha_k$ *and* $u = u_k$ *fixed (i.e. we keep getting* $\rho_{k+1} < \xi$*) or*

2. *we have* $\rho_{k+1} \geq \xi$, $\alpha_k \downarrow \alpha$, $u_k \to u$ *where* $u \notin \arg\min \{V_\alpha(w) \mid w \in S^n\}$.

Then

$$\liminf_{k\to\infty} \rho_{k+1} \geq 1.$$

Proof. We have

$$\liminf_{k\to\infty} \rho_{k+1} = \min\left\{ \lim_{m\to\infty} \rho_{k_m+1} \mid \{\rho_{k_m+1}\}_{m=0}^{\infty} \text{ is a convergent subsequence of } \{\rho_{k+1}\}_{k=0}^{\infty} \right\}.$$

For 1 apply Lemma 4 to an arbitrary convergent subsequence $\{\rho_{k_m+1}\}_{m=0}^{\infty}$ to obtain an (also convergent) subsequence $\left\{\rho_{k_p+1}\right\}_{p=0}^{\infty}$ with

$$\lim_{m\to\infty} \rho_{k_m+1} = \limsup_{p\to\infty} \rho_{k_p+1} \geq 1 \tag{12}$$

from which the result follows.

For any subsequence $\{\rho_{k_m+1}\}_{m=0}^{\infty}$ to obtain an (also convergent) subsequence $\left\{\rho_{k_p+1}\right\}_{p=0}^{\infty}$ with $\left\{ d_{k_p} := \frac{u_{k_p+1}-u_{k_p}}{\|u_{k_p+1}-u_{k_p}\|} \right\}$ converging to d. For 2 we use $\rho_{k+1} \geq \xi$ and (7) to deduce that

$$\begin{aligned} \frac{V_{\alpha_{k_p}}\left(u_{k_p}+t_p d_p\right) - V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p} &\leq \xi \left[\frac{\beta_{k_p} - V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p} \right] \\ &= \xi \left[\frac{\underline{V}_{k_p+1}\left(u_{k_p}+t_p d_p\right) - \underline{V}_{k_p}\left(u_{k_p}\right)}{t_p} \right]. \end{aligned} \tag{13}$$

Now as $u_k \to u \notin \arg\min \{V_\alpha(w) \mid w \in S^n\}$ for k large there exists a descent direction for $V_\alpha(\cdot)$ at u_k. As $\underline{V}_k(u_k) = V_\alpha(u_k)$ and $\underline{V}_k(\cdot)$ minorizes $V_\alpha(\cdot)$ there must exist a greater descent in the same direction for $\underline{V}_k(\cdot)$ at u_k. As u_{k_p+1} solves (3) we have $d_{k_p} := \frac{u_{k_p+1}-u_{k_p}}{\|u_{k_p+1}-u_{k_p}\|}$ in the direction of maximal descent of $\underline{V}_{k_p}(\cdot)$ at u_{k_p} and so there exists $\delta > 0$ such that

$$\underline{V}_{k_p+1}\left(u_{k_p}+t_p d_p\right) - \underline{V}_{k_p}\left(u_{k_p}\right) \leq -\delta t_p$$

for p sufficiently large. Thus (13) and the subgradient inequality for $u \mapsto V_{\alpha_{k_p}}(\cdot)$ at u_{k_p} gives (for p large)

$$V'_{\alpha_{k_p}}\left(u_{k_p}, d_p\right) \leq \frac{V_{\alpha_{k_p}}\left(u_{k_p}+t_p d_p\right) - V_{\alpha_{k_p}}\left(u_{k_p}\right)}{t_p} \leq -\xi\delta < 0.$$

Using (11) we have

$$V'_\alpha(u,d) = \limsup_p V'_{\alpha_{k_p}}\left(u_{k_p}, d_p\right) - \xi\delta < 0.$$

We may now apply Lemma 3 to get (12) again. ■

The following Lemma is an adaptation of Lemma 2 of [5]. This kind of estimate is essential for the proof of convergence of a trust region method and is usually provided by the Cauchy point analysis which is valid only when we have a smooth function that we are modeling. As we have a piecewise affine model function an alternative approach is required. Let $p_{\alpha_k}(u)$ be the projection of u onto the solution set $S_{\alpha_k} \subseteq S^n$ for minimizers of $u \mapsto V_{\alpha_k}(u)$ and denote $v_k^* := \min_{u\in S^n} V_{\alpha_k}(u)$.

Lemma 5 *For a sequence* $\{\rho_{k,l}\}$ *of inner iterates for which* $\rho_{k,l} < \xi$ *for all* l *and solutions* $\{u_{k,l}\}$ *to the problem (3) we have the estimate*

$$\underline{V}_{k,l}(u_k) - \underline{V}_{k,l}(u_{k,l}) \geq \min\left(\frac{\Delta_{k,l}}{\|u_k - p_{\alpha_k}(u_k)\|_\infty}, 1 \right) \left(V_{\alpha_k}(u_k) - v_k^*\right). \tag{14}$$

Proof. An adaptation of Lemma 2 of [5], see appendix for proof. ■

The finally results we require to analyze the trust region loop is an adaptation of Lemma 4 of [5]. As the proof is runs in an entirely parallel fashion we relegate this to an Appendix.

Lemma 6 *Assume $X(\bar{\alpha})$ is bounded and take $\eta \in (\xi, 1)$. Let k, l_1 be an index such that u_{k,l_1} fails the test $\rho_{k,l_1} \geq \xi$. Then either the subsequent sequence of trust region iterates (with α_k fixed) eventually yields u_{k,l_2} satisfying $\rho_{k,l_2} \geq \xi$ or there is an index $l_2 > l_1$ for which*

$$V_{\alpha_k}(u_k) - \underline{V}_{k,l_2}(u_{k,l_2}) \leq \eta \left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_1}(u_{k,l_1}) \right]. \tag{15}$$

Proof. Straight adaptation of Lemma 4 of [5], see Appendix. ■

We are now able to show that when we do not possess the correct value for $v(SD)$ either a multiplier descent is found or an upper bound is obtained in a finite number of iterations.

Proposition 2 *We have only three possible outcomes in the algorithm SDTR for a loop involving the trust region strategy for a fixed α_k.*

1. *Suppose $u_k \notin S_{\alpha_k}$ then the trust region loop in SDTR for a fixed α_k can only be executed a finite number of times terminating with $\rho_{k,l} > \xi$ and acceptance of a new multiplier.*
2. *Suppose $u_k \in S_{\alpha_k}$.*
   (a) *When $\alpha_k \neq v(SD)$ then the trust region loop in SDTR for a fixed α_k must terminate after a finite number of iteration in step 3(a) of STDR giving α_k an upper bound for $v(SD)$ and a decrease of $\overline{\alpha}$ resulting.*
   (b) *When $\alpha_k = v(SD)$ then $\underline{V}_{k,l_k}(u_{k,l_k}) \to 0 = V_{\alpha_k}(u_k)$, monotonically. In the case when we have a pure integer program and are able to add distinct solution each time we solve $V_{\alpha_k}(u_{k,l_k})$ then in finite number of iterations $\underline{V}P_{k,l}(u_k) = 0$.*

Proof. In all cases we gave $V_{\alpha_k}(u_k) \geq 0$. Suppose that $\rho_{k,l} < \xi$ for all l then applying Lemma 6 recursively we have a set of indices l_p such that

$$\begin{aligned} V_{\alpha_k}(u_k) - \underline{V}_{k,l_{k,p}}(u_{k,l_p}) &\leq \eta \left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_{p-1}}(u_{k,l_{p-1}}) \right] \\ &\leq \eta^2 \left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_{p-2}}(u_{k,l_{p-2}}) \right] \\ &\vdots \\ &\leq \eta^p \left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_0}(u_{k,l_0}) \right] \to_{p\to\infty} 0. \end{aligned} \tag{16}$$

When $u_k \notin S_{\alpha_k}$ we have some $\varepsilon > 0$ such that $\|u_k - p_{\alpha_k}(u_k)\|_\infty \geq \varepsilon$ and $V_{\alpha_k}(u_k) - v_k^* \geq \varepsilon$. Using (14) we have

$$\min\left(\frac{\Delta_{k,l_p}}{\|u_k - p_{\alpha_k}(u_k)\|_\infty}, 1 \right) (V_{\alpha_k}(u_k) - v_k^*) \to_{p\to\infty} 0$$

implying $\Delta_{k,l_p} \to 0$. Now apply Corollary 1 part 1 to extract a subsequence of $\{\rho_{k,l_p}\}_{p=0}^{\infty}$ that tends to 1. This implies that there exists p such that $\rho_{k,l_p} \geq \xi > 0$, a contradiction.

In the case that $u_k \in S_{\alpha_k}$ then there does not exist any descent for $u \mapsto V_{\alpha_k}(u)$ at $u = u_k$. In particular we have $v_k^* = V_{\alpha_k}(u_k) \geq 0$. Consequently the problem (3) cannot generate a descent that passes the test $\rho_k \geq \xi$. Using (16) we observe that

$$v_k^* - \underline{V}_{k,l_{k,p}}(u_{k,l_p}) \to_p 0. \tag{17}$$

As we add a new subgradient each time when solving $V_{\alpha_k}(u_{k,l})$ to calculate $\rho_{k,l}$ the model function $u \mapsto \underline{V}_{k,l_k}(u)$ is monotonically increasing and hence convergent to a finite convex function. As the trust region size is monotonically non-increasing we have $\{\underline{V}_{k,l_k}(u_{k,l_k})\}_k$ monotonically non-decreasing. Consequently using (17) we deduce we have a subsequence converging to zero so for whole sequence

$$\underline{V}_{k,l_k}(u_{k,l_k}) \uparrow v_k^*.$$

When $V_{\alpha_k}(u_k) = v_k^* > 0$ (or equivalently $v(SD) < \alpha_k$, the case when $v(SD) \neq \alpha_k$) we find after a finite number of iterations $\beta_{k,l_k} = \underline{V}_{k,l_{k,p}}(u_{k,l_k}) > 0$. We are thus in case 3(a) of the algorithm SDTR after a finite number of iterations. Alternatively $V_{\alpha_k}(u_k) = v_k^* = 0$ in which case $\alpha_k = v(SD)$ and $\underline{V}_{k,l_k}(u_{k,l_k}) \leq 0$. This corresponds to case in part 2b of the proposition.

Suppose we have a pure IP and we add a new subgradient each time when solving $V_{\alpha_k}(u_{k,l})$ to calculate $\rho_{k,l}$. As $u \mapsto V_{\alpha_k}(u)$ is polyhedral ($X(\alpha_k)$ contains a finite set of points) we must add all extremal subgradients after a finite number of iterations l. As $u_k \in S_{\alpha_k}$ we have

$$0 \in \partial V_{\alpha_k}(u_k) = \operatorname{co}\{u_k(Ax_j - b) \mid x_j \in X_{k,l}\}$$

but as $\underline{V}_{k,l}(u) = \max\{u(Ax_j - b) \mid x_j \in X_{k,l}\}$ we also have

$$0 \in \partial \underline{V}_{k,l}(u_k) = \operatorname{co}\{u_k(Ax_j - b) \mid x_j \in X_{k,l}\}$$

and so u_k is a local (and hence global) minimum of $u \mapsto \underline{V}_{k,l}(u)$. Note that

$$0 \leq \underline{V}_{k,l}(u_k) = u_k(Ax_k - b) \leq \min_{u \in S^n \cap B} \underline{V}_{k,l}(u) = \beta_{k,l}$$

as $u_k(Ax_k - b) = V_{\alpha_k}(u_k) \geq 0$. Hence $\underline{V}_{k,l}(u_k) = 0$ in a finite number of iterations. ■

We are now in a position to prove formally that our model algorithm converges.

Theorem 2 *The algorithm SDTR terminates either after a finite number of iterations with either $|\overline{\alpha} - \underline{\alpha}| \leq \varepsilon$ or for some k we have $\alpha_k = v(SD)$ and $\underline{V}_{k,l_k}(u_{k,l_k}) \to 0 = V_{\alpha_k}(u_k)$, monotonically. In the case when we have a pure IP and we add a new subgradient each time when solving $V_{\alpha_k}(u_{k,l})$ to calculate $\rho_{k,l}$ the latter case cannot occur i.e. we always terminate after a finite number of iterations with $|\overline{\alpha} - \underline{\alpha}| \leq \varepsilon$.*

Proof. Whenever $[\overline{\alpha}, \underline{\alpha}]$ is updated we decrease the length of the interval of uncertainty $[\overline{\alpha}, \underline{\alpha}]$ by at least a constant factor. Thus finite termination is assured unless we have an infinite cycle in the trust region loop with fixed interval $[\overline{\alpha}, \underline{\alpha}]$. Proposition 2 indicates that this can only be associated with a sequence of decreasing $\{\alpha_k\}$ and a sequence of acceptable descents (i.e. $\rho_{k,l} \geq \xi$) which does not terminate or the case is that for some k we have $\alpha_k = v(SD)$ and $\underline{V}_{k,l_k}(u_{k,l_k}) \to 0 = V_{\alpha_k}(u_k)$, monotonically. This later case cannot occur when we have a pure IP.

Suppose now that there is a sequence of decreasing $\{\alpha_k\}$ and a sequence of acceptable descents (i.e. $\rho_{k,l} \geq \xi$) which does not terminate. Thus there exists a sequence $\{l_k\}$ such that $u_{k+1} = u_{k,l_k}$ and between iteration $(k,1)$ and (k,l_k) we have $\alpha_{k+l} = \alpha_k$ fixed. Thus by Lemma 5

$$\begin{aligned}
&\underline{V}_{k,1}(u_k) - \underline{V}_{k,l_k}(u_{k,l_k}) \\
&\qquad = \sum_{l=1}^{l_k} \underline{V}_{k,l}(u_k) - \underline{V}_{k,l}(u_{k,l}) \\
&\qquad \geq \sum_{l=1}^{l_k} \min\left(\frac{\Delta_{k,l}}{\|u_k - p_{\alpha_k}(u_k)\|_\infty}, 1\right)(V_{\alpha_k}(u_k) - v_k^*)
\end{aligned}$$

where $u_{k+1} = u_{k,l_k}$ and $\underline{V}_{k+1,0}(u_{k+1}) = \underline{V}_{k,l_k}(u_{k,l_k}) \geq 0$ (as we are in step 3 pf STDR). The model function $\underline{V}_{k,l_k}(\cdot)$ is not carried to the next stage but α_k is decrease to α_{k+1} and we prune subgradients which decrease the new model function to the new initial model $\underline{V}_{k+1,1}(\cdot)$. Hence $\underline{V}_{k,l_k}(u_{k,l_k}) - \underline{V}_{k+1,1}(u_{k+1}) \geq 0$. Thus

$$\begin{aligned}
&\underline{V}_{k,1}(u_k) - \underline{V}_{k+1,1}(u_{k+1}) = \\
&\qquad \underline{V}_{k,1}(u_k) - \underline{V}_{k,l_k}(u_{k,l_k}) + \underline{V}_{k,l_k}(u_{k,l_k}) - \underline{V}_{k+1,1}(u_{k+1}) \\
&\qquad \geq \sum_{l=1}^{l_k} \min\left(\frac{\Delta_{k+l}}{\|u_k - p_{\alpha_k}(u_k)\|_\infty}, 1\right)(V_{\alpha_k}(u_k) - v_k^*).
\end{aligned}$$

For $M = \sum_{k=0}^{K} l_k$ (as we remain in step 3 of SDTR with $\underline{V}_{K,1}(u_K) \geq 0$) we have

$$\begin{aligned}
+\infty > \underline{V}_{0,1}(u_0) &\geq \underline{V}_{0,1}(u_0) - \underline{V}_{K,1}(u_K) \\
&= \sum_{k=0}^{K-1} \left[\underline{V}_{k,1}(u_k) - \underline{V}_{k+1,1}(u_{k+1})\right] \\
&\geq \sum_{k=0}^{K-1} \sum_{l=0}^{l_k} \min\left(\frac{\Delta_{k+l}}{\|u_k - p_{\alpha_k}(u_k)\|_\infty}, 1\right)\left(V_{\alpha_k}(u_k) - v_k^*\right) \\
&= \sum_{j=0}^{M} \min\left(\frac{\Delta_j}{\|u_j - p_{\alpha_j}(u_j)\|_\infty}, 1\right)\left(V_{\alpha_j}(u_j) - v_j^*\right)
\end{aligned}$$

where $p_{\alpha_j}(u)$ be the projection of u onto the solution set $S_{\alpha_j} \subseteq S^n$ for minimizers of $u \mapsto V_{\alpha_j}(u)$ and $v_j^* := \min_{u \in S^n} V_{\alpha_j}(u)$. Hence

$$\min\left(\frac{\Delta_j}{\|u_j - p_{\alpha_j}(u_j)\|_\infty}, 1\right)\left(V_{\alpha_j}(u_j) - v_j^*\right) \to_{j\to\infty} 0.$$

Next note that each time we accept a subgradient (as we have obtained sufficient descent) we decrease the interval $[\underline{\alpha}, \alpha_k]$. Thus we have $\alpha_j \downarrow \underline{\alpha}$ as $j \to \infty$. As $\min_{u \in S^n} V_{\underline{\alpha}}(u) < 0$ and we assume we remain in step 3 pf SDTR we have $V_{\alpha_j}(u_j) \geq 0$ and hence $\|u_j - p_{\alpha_j}(u_j)\|_\infty \geq \delta > 0$ for some δ and hence there is some $\epsilon > 0$ such that $V_{\alpha_j}(u_j) - v_j^* \geq \epsilon > 0$. Consequently we must have $\Delta_j \to 0$.

Consider the first case when we have a MIP. Next note that each time we accept a subgradient (as we have obtained sufficient descent) we reduce the interval $[\underline{\alpha}, \alpha_k]$. Thus we have $\alpha_j \downarrow \underline{\alpha}$ as $j \to \infty$ and we may apply Corollary 1 to deduce that $\liminf_j \rho_j \geq 1$. But this implies that there exists a J for which $\rho_j \geq \frac{3}{4}$ for $j \geq J$ forcing $\Delta_{j+l} = \min\left\{2\Delta_{j+l-1}, \overline{\Delta}\right\}$. Eventually we must have $\Delta_j = \overline{\Delta} > 0$ a contradiction.

In the case of a pure IP (no continuous variable) we note that as $\alpha_j \downarrow 0$ and $X(\alpha_j) := \{x \in X \mid cx \leq \alpha_j\}$ for j sufficiently large the discrete components of $X(\alpha_j)$ do not change. As we have a pure IP the function $V_{\alpha_j}(\cdot)$ must be constant and equal to $V_{\underline{\alpha}}(\cdot)$ for j sufficiently large. Hence for j sufficiently large $V_{\alpha_j}(\cdot) = V_{\underline{\alpha}}(\cdot)$. Consequently we may apply Corollary 1 with $\alpha_k = \underline{\alpha}$ constant (and hence $X_k(\underline{\alpha})$ does not shed any additional elements) to deduce that $\liminf_j \rho_j \geq 1$.

Thus we cannot have an infinite loop of acceptable descents without an update of a lower or upper bound. After a finite number of iteration we must have $|\overline{\alpha} - \underline{\alpha}| \leq \varepsilon$. ■

5 Implementation and Numerical Results

Here we describe our test data, provide details of the algorithm implementation and parameter settings used in our numerical experiments, describe the algorithm based on the ConicBundle solver [10] that we used as a benchmark, and give the experimental results.

5.1 Test Data

We use two sets of problems as our testbed. Set (A) consists of 151 Multiple Choice Multidimensional Knapsack problems (MCMDK) belonging to three of the hardest classes of problems presented in [9], specifically the G-L-D(SU), G-C(L)-S and G-C(L)-D(S) instances (refer to Table 3, for example, in [9]). We only used problems where the LP solution is not optimal, and label them by number using the prefix GLDSU, GCLS and GCLDS for instances from each of the three classes respectively. Set (B) consists of 13 MCMDK problems available from [11]. Some of these problems remain unsolved. For all problems, we dualize only the inequality constraints, keeping the equality "multiple choice" constraints in X. Note that in their original form these are maximization problems, with all variables binary and all objective coefficients non-negative. To ensure results are in keeping with the minimization form of the problem used throughout the paper, we have converted all instances into minimization problems by multiplying their objective functions by -1. Note also that zero must therefore be an upper bound on the value of the IP, and hence on its SD.

5.2 Algorithm Details and Parameter Settings

We implemented our algorithm in C++ using IBM ILOG CPLEX 12.1 for the solution of the subproblems. We used the following parameter settings: $\varepsilon = 0.05$, $\Delta_1 = 1$, $\gamma = 0.5$, $\bar{\gamma} = 2$ and $\xi = 10^{-4}$. The initial upper bound on the SD is taken to be $\bar{\alpha} := 0$, and the lower bound $\underline{\alpha}$ is initialized to the LP relaxation of the instance. The initial value of α is set to $\alpha_1 := 0.9\underline{\alpha} + 0.1\bar{\alpha} = 0.9\underline{\alpha}$. The initial multipliers u_1 are set to the LP dual values for the knapsack constraints.

As noted in Section 2, we also use tolerances for any test used to deduce a new bound on the value of the SD. In particular, in Step 2, we only deduce that $V_{\alpha_k}(u_k) < 0$ if it is in fact strictly less than -10^{-6}. For our test data, no such tolerance is required in the deduction of a new upper bound in Step 2, since all variables and data in the primal IP are integer, with the two exceptions being instances I05 and I06 in Set (B). These exceptions did not give rise to any numerical problems, however in general the use of tolerance in deduction of an upper bound should be considered. In (alternative) Step 3(a), we are very aggressive in pursuing upper bounds, accepting $\beta^* \geq 0$ if it is at least -10^{-9}, and only asking that $\underline{V}_k(\hat{u}) > -10^{-9}$ before concluding that α_k is an upper bound.

This conservative strategy in accepting lower bound updates obviously introduces the possibility of failing to identify a new lower bound when one is in fact available. Initially this could lead to unnecessary search for a better multiplier. Worse, it may happen that as the multipliers approach optimality, these numerical safeguards will cause the algorithm to fail to compute the surrogate dual value, and to stall. Taking a more aggressive approach to the upper bound mitigates this risk, albeit introducing a new risk: that the SD upper bound finally reported by the algorithm is not valid. In our computational experiments, we found that on balance these risks were worth taking. Furthermore, our primary use for the algorithm is the computation of the lower bound, for which we err on the "safe side".

We test two variants of our algorithm, differing according to how the value iterate is updated during the algorithm. In the first or "basic" variant, which we denote by (SDTRB), the value iterate is kept constant, with $\alpha_{k+1} := \alpha_k$ in Step 3(b), and is updated using the bisection rule, $\alpha_{k+1} := (\underline{\alpha} + \bar{\alpha})/2$, in Steps 2 and (alternative) Step 3(a) (both cases in which the value iterate is updated after a new bound has been found for it). The second variant, which we denote by (SDTR), tests the opposite extreme, in which the value iterate is changed at every iteration: in Steps 2 and (alternative) Step 3(a), we set $\alpha_{k+1} := 0.9\underline{\alpha} + 0.1\bar{\alpha}$, and in Step 3(b) we set $\alpha_{k+1} := 0.9\underline{\alpha} + 0.1\alpha_k$. The motivation is to not be too ambitious for too long, since for most of our test data the surrogate dual value is not very far from the LP value.

To evaluate the performance of our trust region algorithm, we compare it with the nonsmooth optimization solver ConicBundle available from [10]. In doing so we solve a modified form of $D(\alpha)$ for each value of α tested: ConicBundle can only handle box constraints, thus instead of solving $D(\alpha)$, which minimizes over $S^n = \{u \geq 0 | eu = 1\}$, we solve $D'(\alpha)$ defined by

$$v(D'(\alpha_k)) := \min_{u \in S^{n'}} V_\alpha(u),$$

where $S^{n'} := \{0 \leq u \leq 1\}$. As a consequence $v(D'(\alpha))$ can never be positive ($V_\alpha(0) = 0$). We have implemented the following simple algorithm based on ConicBundle.

Algorithm CBSD

Step 0. set ε, the stopping tolerance parameter
set $\bar{\alpha}$ to some upper bound on $v(SD)$ (possibly $+\infty$)
set $\underline{\alpha}$ to some lower bound on $v(SD)$ (usually the LP relaxation)
set $k := 1$

Step 1. **if** $|\bar{\alpha} - \underline{\alpha}| \leq \varepsilon$ **then** stop: $v(SD)$ has been found to within the tolerance
else set $\alpha_k := (\underline{\alpha} + \bar{\alpha})/2$
endif

Step 2. Use ConicBundle to solve $D'(\alpha_k)$ and thus compute $v(D'(\alpha_k))$
if $v(D'(\alpha_k)) < 0$ **then** /* α_k must be a lower bound */
update $\underline{\alpha} := \alpha_k$
else /* it must be that $v(D'(\alpha_k)) = 0$ and so α_k is an upper bound */
update $\bar{\alpha} := \alpha_k$
endif

$k := k + 1$
go to Step 1

Remark 3 *ConicBundle accumulates subgradients of the function $u \mapsto V_\alpha(u)$ building a model function but it does not provide to the user the option to selectively eliminate subgradients that are no longer valid. When α is increased to α', say, (as occurs when the lower bound $\underline{\alpha}$ is updated), we can maintain the model, since for all $x \in X(\alpha)$ we also have $x \in X(\alpha')$. However, when α is reduced (as occurs when the upper bound $\bar{\alpha}$ is updated) we need to reset ConicBundle's model.*

Remark 4 *In SDTR and SDTRB we prematurely quit the solver of $P(\alpha, u)$ when we can deduce the sign of $V_\alpha(u)$. In this case we use the best feasible solution $\hat{x}$ of $P(\alpha, u)$ and the valid inequality $u^T(A\hat{x} - b) \geq 0$ to improve our model. This may not necessary be a subgradient as we have not solved $P(\alpha, u)$ to optimality. However, in practice CPLEX heuristics work well. Since the correct sign of $V_\alpha(u)$ is enough to protect the integrity of our algorithm, ignoring proof of optimality is an attractive strategy to improve performance. Unfortunately this trick does not work well with CBSD as inaccurate function evaluations appear to affect the stability of ConicBundle.*

5.3 Performance of SDTR, SDTRB and CBSD in Solving Surrogate Dual Problems

In Tables 1, 2 and 3 we give results for the three algorithms on each of the three classes of problems that comprise Set (A). The corresponding results for Set (B) are given in Table 4. Each table reports the value of the LP relaxation of the IP, ("LP Value"), the final lower bound on the value of its surrogate dual found by each algorithm (the value of $\underline{\alpha}$ when the algorithm terminates), ("SD Value"), the optimal value of the IP ("IP Value"), the time each algorithm requires to run on the instance in seconds, ("SD Time"), the number of times $P(\alpha, u)$ is solved in the course of the algorithm, the number of descent steps in the multiplier space made by ConicBundle, ("D-Steps") and the number of LP solves required to minimize the model function in Step 3 during the course of the algorithm, for each of SDTR and SDTRB ("M-Calls"). We report the value of all dual bounds accurate to three decimal places, and run times to two decimal places. Note that since each algorithm terminates when its upper and lower bound are within 0.05 of each other, there is some variation in the SD Value entry obtained by each algorithm. Analysis shows that no one algorithm tends to give higher or lower values; each returns a value that is on average not worse than the maximum of the three by about the same amount.

In all cases, SDTR and SDTRB run consistently faster than CBSD. This effect is most evident in Tables 1 and 4, which report results for instances that take longer to solve (particularly in the latter case). In the former case, SDTRB is on average faster than SDTR, with the average run time of the former just over half that of the latter. In the case of Table 4, which has the hardest problems, the order is reversed: average run time of SDTR is just over a third of that of SDTRB. Analysis in which the ratio of time needed for CBSD and SDTRB to that needed for SDTR is computed for each instance, and then averaged, shows that on average CBSD takes just over 322 times as long as SDTR to solve each instance in Table 4, while SDTRB takes on average just over twice as long. We conclude that the choice to interleave search in the multiplier space with search in the value space (as in SDTR) can be helpful in speeding up solution, but that the dominant effect is that of our bundle trust approach. Indeed SDTRB is identical to CBSD except for the method used to search in the multiplier space, and it is clear that the former shows far better computational performance. However the relative improvement due to interleaving the search is also significant on the hardest class of problems.

5.4 Using the Surrogate Dual Bound to Accelerate IP Solution

In this section, we demonstrate the potential value of the bound that can be obtained from the surrogate dual in solving the original integer programs, in a simple way. We first compute (a lower bound on) the surrogate dual value $v(SD)$, (as in the previous section) and then solve the original IP with the additional constraint $cx \geq v(SD)$ using CPLEX. The results of this are compared with those for solving the original IP with CPLEX. To make the comparison fair, and since CPLEX may make use of an additional constraint in clever ways, such as in preprocessing, we also check that adding the constraint $cx \geq v(LP)$, where $v(LP)$ denotes the value of the relaxation, doesn't significantly change the solution times for CPLEX. Thus we can be sure that any improvement is as a result of the stronger

Table 1: Results for Set (A), class G-L-D(SU). P-Calls indicates the number of $P(\alpha, u)$ problems solved and M-Calls the number of LP's solved for SDTR and SDTRB. D-Steps indicates the number of descent steps made in the multiplier space for CBSD. SD Time gives the run times for each algorithm in seconds.

Instance	LP	SD Value (Final $\underline{\alpha}$)			IP	SD Time (s)			P-Calls			D-Steps	M-Calls	
	Value	CBSD	SDTRB	SDTR	Value	CBSD	SDTRB	SDTR	CBSD	SDTRB	SDTR	CBSD	SDTRB	SDTR
GLDSU41	-172.258	-170.029	-170.026	-170.002	-170	0.06	0.03	0.05	46	20	31	12	14	23
GLDSU42	-176.835	-170.013	-170.031	-170.005	-170	0.08	0.05	0.06	70	30	41	19	28	13
GLDSU43	-180.305	-180.041	-180.023	-180.008	-170	0.49	0.04	0.04	273	19	25	23	29	17
GLDSU44	-184.518	-180.013	-180.013	-180.034	-180	0.94	0.03	0.04	568	18	23	19	19	16
GLDSU45	-188.897	-180.042	-180.005	-180.001	-180	0.09	0.03	0.05	71	18	37	14	19	15
GLDSU46	-193.000	-190.031	-190.022	-190.004	-190	0.06	0.02	0.05	52	19	26	12	18	17
GLDSU47	-197.000	-190.037	-190.036	-190.006	-190	0.05	0.04	0.04	46	25	28	12	22	15
GLDSU48	-201.000	-200.019	-200.046	-200.002	-200	1.03	0.03	0.04	609	17	25	22	15	19
GLDSU49	-205.000	-200.020	-200.035	-200.012	-200	0.09	0.03	0.04	70	16	23	17	20	16
GLDSU50	-209.000	-200.020	-200.020	-200.007	-200	0.10	0.02	0.03	85	19	21	18	21	19
GLDSU51	-213.000	-210.010	-210.005	-210.020	-210	0.13	0.03	0.03	98	16	24	18	24	16
GLDSU52	-216.000	-210.015	-210.009	-210.036	-210	0.08	0.02	0.04	64	19	29	16	23	16
GLDSU54	-224.000	-220.008	-220.019	-220.016	-220	0.05	0.02	0.04	53	16	28	21	20	14
GLDSU55	-228.000	-220.012	-220.029	-220.009	-220	0.07	0.02	0.02	56	12	11	16	13	11
GLDSU56	-232.000	-230.037	-230.006	-230.028	-230	0.07	0.03	0.04	65	13	24	15	18	10
GLDSU57	-236.000	-230.008	-230.008	-230.042	-230	0.04	0.02	0.03	40	12	21	12	18	8
GLDSU59	-244.000	-239.979	-240.045	-240.001	-240	0.06	0.01	0.02	58	12	19	18	12	5
GLDSU60	-247.000	-240.049	-240.005	-240.010	-240	0.02	0.02	0.02	28	12	20	12	16	7
GLDSU61	-251.000	-250.020	-250.027	-250.048	-250	0.14	0.01	0.04	129	10	30	13	5	3
GLDSU62	-255.000	-250.032	-250.028	-250.047	-250	0.02	0.01	0.02	24	11	17	11	3	3
GLDSU63	-259.000	-250.025	-250.039	-250.050	-250	0.04	0.01	0.01	39	10	13	16	3	3
GLDSU64	-263.000	-260.027	-260.021	-260.003	-260	0.13	0.02	0.02	130	13	17	17	14	10
GLDSU65	-267.000	-260.028	-260.012	-260.007	-260	0.04	0.02	0.02	35	12	15	15	15	5
GLDSU66	-271.000	-270.029	-270.021	-270.007	-270	0.02	0.03	0.03	29	12	13	13	15	7
GLDSU67	-275.000	-270.044	-270.049	-270.025	-270	0.03	0.01	0.00	27	9	4	10	0	0
GLDSU68	-279.000	-270.035	-270.032	-270.047	-270	0.02	0.02	0.04	20	10	38	8	3	3
GLDSU69	-282.000	-280.028	-280.017	-280.023	-280	0.02	0.02	0.02	26	13	22	13	18	8
GLDSU70	-286.000	-280.048	-280.038	-280.049	-280	0.02	0.01	0.03	21	10	29	11	0	0
GLDSU72	-294.000	-290.027	-290.009	-290.009	-290	0.02	0.02	0.02	24	12	18	13	11	4
GLDSU73	-298.000	-290.026	-290.034	-290.046	-290	0.02	0.01	0.01	24	7	9	13	0	0
GLDSU74	-302.000	-300.026	-300.028	-300.047	-300	0.02	0.01	0.03	25	7	37	13	0	0
GLDSU75	-306.000	-300.038	-300.034	-300.048	-300	0.02	0.01	0.02	24	10	26	11	0	0
GLDSU77	-313.000	-310.035	-310.033	-310.049	-310	0.02	0.01	0.04	23	10	41	11	0	0
GLDSU78	-317.000	-310.032	-310.047	-310.047	-310	0.02	0.01	0.03	23	10	36	12	0	0
GLDSU79	-321.000	-320.047	-320.031	-320.049	-320	0.01	0.01	0.03	25	11	28	12	0	0
GLDSU80	-325.000	-320.038	-320.029	-320.047	-320	0.01	0.01	0.04	23	9	37	13	0	0
GLDSU81	-329.000	-320.030	-320.048	-320.049	-320	0.02	0.00	0.03	21	7	30	13	0	0
GLDSU82	-333.000	-330.050	-330.029	-330.049	-330	0.02	0.01	0.05	23	10	41	10	0	0
GLDSU83	-337.000	-330.027	-330.044	-330.050	-330	0.02	0.01	0.03	21	10	34	12	0	0
GLDSU84	-341.000	-340.042	-340.029	-340.047	-340	0.02	0.01	0.03	25	11	28	12	0	0
GLDSU85	-344.000	-340.041	-340.029	-340.047	-340	0.02	0.01	0.02	24	11	20	12	0	0
GLDSU86	-348.000	-340.040	-340.029	-340.046	-340	0.02	0.01	0.05	24	11	40	12	0	0
GLDSU87	-352.000	-350.039	-350.028	-350.046	-350	0.02	0.01	0.04	23	11	37	12	0	0
GLDSU88	-356.000	-350.027	-350.048	-350.050	-350	0.02	0.01	0.02	23	9	18	10	0	0
GLDSU90	-364.000	-360.036	-360.027	-360.046	-360	0.02	0.01	0.01	21	11	6	12	0	0
GLDSU91	-368.000	-360.035	-360.027	-360.046	-360	0.02	0.01	0.04	23	11	39	12	0	0
GLDSU92	-372.000	-370.034	-370.026	-370.045	-370	0.02	0.02	0.04	23	11	37	12	0	0
GLDSU93	-376.000	-370.031	-370.041	-370.048	-370	0.01	0.01	0.04	19	9	39	8	0	0
GLDSU94	-379.000	-370.048	-370.033	-370.049	-370	0.02	0.02	0.02	21	11	20	12	0	0
GLDSU95	-383.000	-380.008	-380.038	-380.048	-380	0.01	0.01	0.04	19	9	41	7	0	0
GLDSU96	-387.000	-380.030	-380.034	-380.046	-380	0.02	0.01	0.04	21	10	43	11	0	0
GLDSU97	-391.000	-390.030	-390.049	-390.049	-390	0.03	0.01	0.02	30	10	27	12	0	0
GLDSU98	-395.000	-390.030	-390.031	-390.048	-390	0.02	0.00	0.02	25	5	30	13	0	0
Average						0.084	0.017	0.032	64.3	12.6	26.7	13.5	8.2	5.7
Std. Dev.						0.192	0.010	0.012	111.7	4.6	9.8	3.4	9.4	7.0

Table 2: Results for Set (A), class G-C(L)-S. P-Calls indicates the number of $P(\alpha, u)$ problems solved and M-Calls the number of LP's solved for SDTR and SDTRB. D-Steps indicates the number of descent steps made in the multiplier space for CBSD. SD Time gives the run times for each algorithm in seconds.

Instance	LP	SD Value (Final $\underline{\alpha}$)			IP	SD Time (s)			P-Calls			D-Steps	M-Calls	
	Value	CBSD	SDTRB	SDTR	Value	CBSD	SDTRB	SDTR	CBSD	SDTRB	SDTR	CBSD	SDTRB	SDTR
GCLS2	-452.000	-451.029	-451.037	-451.049	-451	0.01	0.01	0.03	22	10	26	11	0	0
GCLS5	-454.000	-453.028	-453.036	-453.042	-453	0.02	0.02	0.01	23	10	7	11	0	0
GCLS7	-456.000	-455.027	-455.036	-455.040	-455	0.02	0.02	0.01	24	10	7	11	0	0
GCLS10	-458.000	-457.026	-457.036	-457.038	-457	0.02	0.01	0.01	24	10	7	11	0	0
GCLS12	-460.000	-459.025	-459.035	-459.037	-459	0.02	0.01	0.01	25	10	7	11	0	0
GCLS13	-460.000	-459.025	-459.035	-459.037	-459	0.02	0.02	0.01	25	10	7	11	0	0
GCLS15	-462.000	-461.049	-461.035	-461.035	-461	0.02	0.01	0.01	24	10	7	10	0	0
GCLS17	-464.000	-463.047	-463.034	-463.034	-463	0.02	0.02	0.01	24	10	7	10	0	0
GCLS18	-464.000	-463.047	-463.034	-463.034	-463	0.01	0.01	0.01	24	10	7	10	0	0
GCLS20	-466.000	-465.045	-465.034	-465.032	-465	0.02	0.01	0.01	25	10	7	10	0	0
GCLS22	-468.000	-467.043	-467.034	-467.030	-467	0.02	0.01	0.00	25	10	7	10	0	0
GCLS23	-468.000	-467.043	-467.034	-467.030	-467	0.02	0.01	0.00	25	10	7	10	0	0
GCLS25	-470.000	-469.041	-469.033	-469.049	-469	0.02	0.01	0.00	25	10	6	10	0	0
GCLS28	-472.000	-471.039	-471.033	-471.048	-471	0.02	0.01	0.00	25	10	6	10	0	0
GCLS30	-474.000	-473.037	-473.032	-473.046	-473	0.02	0.01	0.01	25	10	6	10	0	0
GCLS33	-476.000	-475.035	-475.032	-475.044	-475	0.02	0.01	0.01	25	10	6	10	0	0
GCLS35	-478.000	-477.033	-477.032	-477.043	-477	0.02	0.01	0.00	25	10	6	10	0	0
GCLS38	-480.000	-479.031	-479.031	-479.041	-479	0.02	0.01	0.00	25	10	6	10	0	0
GCLS40	-482.000	-481.029	-481.031	-481.040	-481	0.02	0.01	0.01	26	10	6	10	0	0
GCLS41	-482.000	-481.029	-481.031	-481.040	-481	0.02	0.01	0.01	26	10	6	10	0	0
GCLS43	-484.000	-483.027	-483.030	-483.038	-483	0.01	0.01	0.00	26	10	6	10	0	0
GCLS45	-486.000	-485.025	-485.030	-485.036	-485	0.02	0.01	0.00	26	10	6	10	0	0
GCLS46	-486.000	-485.025	-485.030	-485.036	-485	0.02	0.01	0.00	26	10	6	10	0	0
GCLS48	-488.000	-487.047	-487.030	-487.035	-487	0.02	0.01	0.00	25	10	6	9	0	0
GCLS50	-490.000	-489.043	-489.029	-489.033	-489	0.01	0.01	0.00	25	10	6	9	0	0
GCLS51	-490.000	-489.043	-489.029	-489.033	-489	0.01	0.01	0.00	25	10	6	9	0	0
GCLS53	-492.000	-491.039	-491.029	-491.031	-491	0.02	0.01	0.00	25	10	6	9	0	0
GCLS56	-494.000	-493.035	-493.029	-493.030	-493	0.02	0.01	0.00	25	10	6	9	0	0
GCLS58	-496.000	-495.031	-495.028	-495.028	-495	0.02	0.01	0.00	25	10	6	9	0	0
GCLS61	-498.000	-497.027	-497.028	-497.027	-497	0.02	0.01	0.00	25	10	6	9	0	0
GCLS63	-500.000	-499.023	-499.027	-499.010	-499	0.02	0.01	0.00	25	10	6	9	0	0
GCLS64	-500.000	-499.023	-499.027	-499.010	-499	0.01	0.01	0.00	25	10	6	9	0	0
GCLS66	-502.000	-501.020	-501.027	-501.048	-501	0.02	0.01	0.00	25	10	5	9	0	0
GCLS68	-504.000	-503.016	-503.027	-503.046	-503	0.02	0.01	0.01	25	10	5	9	0	0
GCLS69	-504.000	-503.016	-503.027	-503.046	-503	0.02	0.01	0.01	25	10	5	9	0	0
GCLS71	-506.000	-505.012	-505.026	-505.045	-505	0.02	0.01	0.01	25	10	5	9	0	0
GCLS73	-508.000	-507.008	-507.026	-507.043	-507	0.02	0.01	0.01	25	10	5	9	0	0
GCLS74	-508.000	-507.008	-507.026	-507.043	-507	0.01	0.01	0.00	25	10	5	9	0	0
GCLS76	-510.000	-509.004	-509.025	-509.041	-509	0.02	0.01	0.00	25	10	5	9	0	0
GCLS79	-512.000	-511.031	-511.025	-511.039	-511	0.03	0.01	0.00	32	10	5	14	0	0
GCLS81	-514.000	-513.031	-513.049	-513.037	-513	0.02	0.01	0.00	32	9	5	14	0	0
GCLS84	-516.000	-515.031	-515.048	-515.036	-515	0.03	0.01	0.00	33	9	5	14	0	0
GCLS86	-518.000	-517.031	-517.048	-517.034	-517	0.03	0.00	0.00	33	9	5	14	0	0
GCLS89	-520.000	-519.031	-519.047	-519.032	-519	0.02	0.00	0.00	33	9	5	14	0	0
GCLS91	-522.000	-521.031	-521.046	-521.030	-521	0.02	0.01	0.00	30	9	5	14	0	0
GCLS92	-522.000	-521.031	-521.046	-521.030	-521	0.02	0.01	0.00	30	9	5	14	0	0
GCLS94	-524.000	-523.031	-523.045	-523.028	-523	0.03	0.01	0.01	29	9	5	14	0	0
GCLS96	-526.000	-525.030	-525.045	-525.027	-525	0.02	0.01	0.01	31	9	5	14	0	0
GCLS97	-526.000	-525.030	-525.045	-525.027	-525	0.03	0.01	0.01	31	9	5	14	0	0
GCLS99	-528.000	-527.030	-527.044	-527.025	-527	0.02	0.01	0.00	31	9	5	14	0	0
Average						0.02	0.01	0.00	26.30	9.80	6.26	10.68	0.00	0.00
Std. Dev.						0.00	0.00	0.01	2.84	0.40	2.92	1.87	0.00	0.00

Table 3: Results for Set (A), class G-C(L)-D(S). P-Calls indicates the number of $P(\alpha, u)$ problems solved and M-Calls the number of LP's solved for SDTR and SDTRB. D-Steps indicates the number of descent steps made in the multiplier space for CBSD. SD Time gives the run times for each algorithm in seconds.

Instance	LP	SD Value (Final $\underline{\alpha}$)			IP	SD Time (s)			P-Calls			D-Steps	M-Calls	
	Value	CBSD	SDTRB	SDTR	Value	CBSD	SDTRB	SDTR	CBSD	SDTRB	SDTR	CBSD	SDTRB	SDTR
GCLDS7	-454.000	-453.028	-453.036	-453.042	-453	0.02	0.01	0.00	25	10	7	11	0	0
GCLDS8	-454.000	-453.028	-453.036	-453.042	-453	0.01	0.01	0.01	26	10	7	11	0	0
GCLDS10	-456.000	-455.027	-455.036	-455.040	-455	0.02	0.01	0.01	27	10	7	11	0	0
GCLDS12	-458.000	-457.026	-457.036	-457.038	-457	0.02	0.01	0.00	25	10	7	11	0	0
GCLDS13	-458.000	-457.026	-457.036	-457.038	-457	0.02	0.01	0.01	28	10	7	11	0	0
GCLDS15	-460.000	-459.025	-459.035	-459.037	-459	0.02	0.02	0.01	30	10	7	11	0	0
GCLDS17	-462.000	-461.049	-461.035	-461.035	-461	0.01	0.01	0.00	25	10	7	10	0	0
GCLDS18	-462.000	-461.049	-461.035	-461.035	-461	0.03	0.01	0.00	31	10	7	10	0	0
GCLDS20	-464.000	-463.047	-463.034	-463.034	-463	0.02	0.01	0.01	29	10	7	10	0	0
GCLDS23	-466.000	-465.045	-465.034	-465.032	-465	0.03	0.01	0.00	35	10	7	10	0	0
GCLDS25	-468.000	-467.043	-467.034	-467.030	-467	0.03	0.01	0.00	31	10	7	10	0	0
GCLDS28	-470.000	-469.041	-469.033	-469.049	-469	0.03	0.01	0.00	35	10	6	10	0	0
GCLDS30	-472.000	-471.039	-471.033	-471.048	-471	0.03	0.01	0.01	31	10	6	10	0	0
GCLDS33	-474.000	-473.037	-473.032	-473.046	-473	0.03	0.01	0.00	31	10	6	10	0	0
GCLDS35	-476.000	-475.035	-475.032	-475.044	-475	0.02	0.01	0.00	26	10	6	10	0	0
GCLDS36	-476.000	-475.035	-475.032	-475.044	-475	0.03	0.01	0.00	35	10	6	10	0	0
GCLDS38	-478.000	-477.033	-477.032	-477.043	-477	0.02	0.01	0.01	31	10	6	10	0	0
GCLDS40	-480.000	-479.031	-479.031	-479.041	-479	0.02	0.01	0.00	26	10	6	10	0	0
GCLDS41	-480.000	-479.031	-479.031	-479.041	-479	0.03	0.01	0.00	35	10	6	10	0	0
GCLDS43	-482.000	-481.029	-481.031	-481.040	-481	0.02	0.01	0.00	33	10	6	10	0	0
GCLDS46	-484.000	-483.027	-483.030	-483.038	-483	0.03	0.01	0.00	38	10	6	10	0	0
GCLDS48	-486.000	-485.025	-485.030	-485.036	-485	0.03	0.01	0.00	33	10	6	10	0	0
GCLDS51	-488.000	-487.047	-487.030	-487.035	-487	0.03	0.01	0.00	37	10	6	9	0	0
GCLDS53	-490.000	-489.043	-489.029	-489.033	-489	0.02	0.01	0.01	32	10	6	9	0	0
GCLDS56	-492.000	-491.039	-491.029	-491.031	-491	0.03	0.01	0.00	32	10	6	9	0	0
GCLDS58	-494.000	-493.035	-493.029	-493.030	-493	0.03	0.01	0.00	32	10	6	9	0	0
GCLDS59	-494.000	-493.035	-493.029	-493.030	-493	0.03	0.01	0.00	37	10	6	9	0	0
GCLDS61	-496.000	-495.031	-495.028	-495.028	-495	0.02	0.01	0.00	32	10	6	9	0	0
GCLDS63	-498.000	-497.027	-497.028	-497.027	-497	0.02	0.01	0.00	26	10	6	9	0	0
GCLDS64	-498.000	-497.027	-497.028	-497.027	-497	0.03	0.01	0.00	37	10	6	9	0	0
GCLDS66	-500.000	-499.023	-499.027	-499.010	-499	0.03	0.01	0.00	32	10	6	9	0	0
GCLDS68	-502.000	-501.020	-501.027	-501.048	-501	0.02	0.01	0.00	26	10	5	9	0	0
GCLDS69	-502.000	-501.020	-501.027	-501.048	-501	0.04	0.01	0.01	37	10	5	9	0	0
GCLDS71	-504.000	-503.016	-503.027	-503.046	-503	0.03	0.01	0.00	32	10	5	9	0	0
GCLDS74	-506.000	-505.012	-505.026	-505.045	-505	0.03	0.01	0.01	37	10	5	9	0	0
GCLDS76	-508.000	-507.008	-507.026	-507.043	-507	0.03	0.01	0.00	32	10	5	9	0	0
GCLDS79	-510.000	-509.004	-509.025	-509.041	-509	0.03	0.01	0.01	37	10	5	9	0	0
GCLDS81	-512.000	-511.031	-511.025	-511.039	-511	0.04	0.01	0.00	40	10	5	14	0	0
GCLDS82	-512.000	-511.031	-511.025	-511.039	-511	0.04	0.01	0.01	44	10	5	14	0	0
GCLDS84	-514.000	-513.031	-513.049	-513.037	-513	0.03	0.01	0.00	38	9	5	14	0	0
GCLDS86	-516.000	-515.031	-515.048	-515.036	-515	0.03	0.01	0.00	33	9	5	14	0	0
GCLDS87	-516.000	-515.031	-515.048	-515.036	-515	0.05	0.01	0.01	56	9	5	14	0	0
GCLDS89	-518.000	-517.031	-517.048	-517.034	-517	0.03	0.01	0.01	41	9	5	14	0	0
GCLDS91	-520.000	-519.031	-519.047	-519.032	-519	0.03	0.01	0.01	32	9	5	14	0	0
GCLDS92	-520.000	-519.031	-519.047	-519.032	-519	0.05	0.01	0.00	56	9	5	14	0	0
GCLDS94	-522.000	-521.031	-521.046	-521.030	-521	0.04	0.00	0.00	41	9	5	14	0	0
GCLDS97	-524.000	-523.031	-523.045	-523.028	-523	0.06	0.01	0.01	57	9	5	14	0	0
Average						0.03	0.01	0.00	34.09	9.83	5.89	10.66	0.00	0.00
Std. Dev.						0.01	0.00	0.00	7.42	0.38	0.75	1.85	0.00	0.00

Table 4: Results for Set (B). P-Calls indicates the number of $P(\alpha, u)$ problems solved and M-Calls the number of LP's solved for SDTR and SDTRB. D-Steps indicates the number of descent steps made in the multiplier space for CBSD. SD Time gives the run times for each algorithm in seconds. A dash "-" in the IP Value column indicates that the optimal value for the instance is not known.

Inst.	LP	SD Value (Final $\underline{\alpha}$)			IP	SD Time (s)			P-Calls			D-Steps	M-Calls	
	Value	CBSD	SDTRB	SDTR	Value	CBSD	SDTRB	SDTR	CBSD	SDTRB	SDTR	CBSD	SDTRB	SDTR
I01	-182.720	-175.577	-175.006	-175.007	-173.0	163.17	0.06	0.06	72634	21	21	33	20	14
I02	-365.570	-364.030	-364.027	-364.047	-364.0	1.34	0.05	0.04	440	18	16	19	16	9
I03	-1629.570	-1624.010	-1624.010	-1624.040	-1602.0	19.92	0.51	0.61	2299	68	61	26	77	46
I04	-3631.360	-3629.010	-3629.010	-3629.010	-3597.0	275.14	1.42	1.55	9600	135	111	25	145	83
I05	-3905.900	-3905.740	-3905.710	-3905.750	-3905.7	0.14	0.06	0.04	58	23	20	16	27	9
I06	-4812.820	-4811.200	-4811.200	-4811.210	-4799.3	0.66	0.07	0.07	228	20	29	30	17	9
I07	-24608.000	-24607.000	-24607.000	-24607.000	-	205.40	8.57	3.89	2522	363	155	24	316	119
I08	-36904.400	-36904.000	-36904.000	-36904.000	-	882.50	14.66	5.40	7892	432	155	29	391	132
I09	-49193.900	-49193.000	-49193.000	-49193.000	-	477.11	18.24	6.99	3712	407	145	27	371	106
I10	-61486.300	-61486.300	-61486.000	-61486.000	-	1550.79	22.81	7.83	3014	404	141	21	368	107
I11	-73797.700	-73797.000	-73797.000	-73797.000	-	4154.75	36.78	7.56	9094	479	111	29	448	95
I12	-86100.500	-86100.500	-86100.000	-86100.000	-	1010.36	41.32	16.09	4432	463	169	21	453	119
I13	-98448.600	-98448.000	-98448.000	-98448.000	-	1552.20	39.94	16.06	4777	392	163	29	377	121
Ave.						791.81	14.19	5.09	9284.8	248.1	99.8	25.3	232.8	74.5
Sd. Dv.						1112.79	15.64	5.51	18541.8	190.0	58.8	4.7	175.0	47.5

dual bound arising from the surrogate dual compared with the LP relaxation. We note that using Lagrangian relaxation with the same X would not provide a bound better than the LP bound since X has the integrality property.

We show results for this on instances in Set (A) in Tables 5, 6 and 7. Note that for the instances in Set (A), the surrogate dual bound often closes a significant proportion of the IP gap, much more so than in Set (B) in the cases for which the optimal value is known. Thus we restrict our results to Set (A). In all cases we use SDTR to compute the surrogate dual value to use in the cut added. The tables show the value of the LP relaxation of each instance, ("LP Value"), the value of the surrogate dual (or rather the final lower bound in it found by the SDTR algorithm), ("SD Value"), the optimal IP value, ("IP Value"), and the proportion of the gap closed by the surrogate dual ("Gap Red."). This is computed as

$$\frac{v(SD) - v(LP)}{v(IP) - v(LP)},$$

where $v(IP)$ denotes the optimal value of the IP. All dual bound values are reported accurate to three decimal places, and the proportion of gap reduced to two decimal places. The last three columns for each table show the run times in seconds for each of the three approaches: running SDTR and then running CPLEX to solve the IP with the additional constraint using the surrogate dual value ("SDTRB + CPLEX"), simply using CPLEX to solve the IP ("CPLEX"), and using CPLEX with the additional constraint using the LP relaxation value ("CPLEX with LP Cut"). Columns 6 and 7 of each table show the breakdown of the time used for the surrogate dual approach into the surrogate dual solution time, ("SDTR"), and the CPLEX solution time, ("CPLEX with SD Cut"). All run times are reported accurate to two decimal places.

It is clear that in all cases, the use of the surrogate dual bound can greatly reduce the time needed to solve the IP. The results on the instances in Tables 6 and 7 which took longer than 100 seconds to solve in CPLEX are particularly noteworthy; the use of the surrogate dual bound appears to reduce the solution times by several orders of magnitude. While the heuristics presented in [9] in general perform better than CPLEX, we note that the relative improvements from those heuristics is nowhere near what we obtain using the surrogate dual value in these experiments.

6 Future Work

Within the general algorithmic framework proposed here, there are clearly many specific choices that could be made differently. For example, whenever choosing a new value iterate, it may be more efficient to revisit previously considered values of α that are not yet resolved, since at these values we expect to have some elements of the model function that are known to correspond to supporting hyperplanes. It may also be worth considering starting at each new value iterate with an initial multiplier iterate that is in some sense "close" to the best identified for nearby values of α.

7 Appendix

Proof of Lemma 5.

Proof. Consider the problem

$$\min_{\tau \in [0,1]} \underline{V}_{k,l}\left(u_k + \tau\left(p_{\alpha_k}\left(u_k\right) - u_k\right)\right) \quad \text{subject to} \quad \left\|\tau\left(p_{\alpha_k}\left(u_k\right) - u_k\right)\right\|_\infty \leq \Delta_{k+l}. \tag{18}$$

Denote the optimal value by $\tau_{k,l}$. As $u_{k,l}$ solves (3) we have

$$\underline{V}_{k,l}\left(u_{k,l}\right) \leq \underline{V}_{k,l}\left(u_k + \tau_{k,l}\left(p_{\alpha_k}\left(u_k\right) - u_k\right)\right) \ .$$

If $\tau = 1$ is feasible then

$$\begin{aligned}
\underline{V}_{k,l}\left(u_{k,l}\right) &\leq \underline{V}_{k,l}\left(u_k + \tau_{k,l}\left(p_{\alpha_k}\left(u_k\right) - u_k\right)\right) \\
&\leq \underline{V}_{k,l}\left(u_k + \ \left(p_{\alpha_k}\left(u_k\right) - u_k\right)\right) = \underline{V}_{k,l}\left(p_{\alpha_k}\left(u_k\right)\right) \\
&\leq V_{\alpha_k}\left(p_{\alpha_k}\left(u_k\right)\right) = v_k^*
\end{aligned}$$

Table 5: Results for solution of IPs for Set (A), class G-L-D(SU).

Instance	LP Value	SD Value (SDTR)	IP Value	Gap Red.	Run Time (s) SDTR	CPLEX with SD Cut	SDTR + CPLEX	CPLEX	Cplex with LP Cut
GLDSU41	-172.258	-170.002	-170	1.00	0.05	0.00	0.05	0.00	0.01
GLDSU42	-176.835	-170.005	-170	1.00	0.06	0.00	0.06	0.01	0.02
GLDSU43	-180.305	-180.008	-170	0.03	0.03	0.06	0.09	0.04	0.04
GLDSU44	-184.518	-180.034	-180	0.99	0.04	0.00	0.04	0.06	0.05
GLDSU45	-188.897	-180.001	-180	1.00	0.05	0.00	0.05	0.15	0.00
GLDSU46	-193.000	-190.004	-190	1.00	0.04	0.00	0.04	0.14	0.13
GLDSU47	-197.000	-190.006	-190	1.00	0.04	0.00	0.04	0.48	0.46
GLDSU48	-201.000	-200.002	-200	1.00	0.05	0.00	0.05	0.01	0.02
GLDSU49	-205.000	-200.012	-200	1.00	0.04	0.00	0.04	1.25	1.37
GLDSU50	-209.000	-200.007	-200	1.00	0.04	0.00	0.04	2.19	2.14
GLDSU51	-213.000	-210.020	-210	0.99	0.02	0.00	0.02	2.01	2.20
GLDSU52	-216.000	-210.036	-210	0.99	0.04	0.00	0.04	4.59	4.64
GLDSU54	-224.000	-220.016	-220	1.00	0.03	0.00	0.03	8.21	8.16
GLDSU55	-228.000	-220.009	-220	1.00	0.01	0.01	0.02	12.65	12.27
GLDSU56	-232.000	-230.028	-230	0.99	0.04	0.00	0.04	5.35	5.94
GLDSU57	-236.000	-230.042	-230	0.99	0.03	0.01	0.04	16.29	15.06
GLDSU59	-244.000	-240.001	-240	1.00	0.02	0.00	0.02	22.63	22.98
GLDSU60	-247.000	-240.010	-240	1.00	0.02	0.00	0.02	25.58	26.32
GLDSU61	-251.000	-250.048	-250	0.95	0.03	0.00	0.03	5.50	7.76
GLDSU62	-255.000	-250.047	-250	0.99	0.02	0.00	0.02	29.74	31.78
GLDSU63	-259.000	-250.050	-250	0.99	0.01	0.00	0.01	37.41	33.55
GLDSU64	-263.000	-260.003	-260	1.00	0.02	0.00	0.02	25.17	29.23
GLDSU65	-267.000	-260.007	-260	1.00	0.02	0.00	0.02	33.92	29.59
GLDSU66	-271.000	-270.007	-270	0.99	0.02	0.00	0.02	9.02	12.99
GLDSU67	-275.000	-270.025	-270	1.00	0.00	0.00	0.00	28.76	32.36
GLDSU68	-279.000	-270.047	-270	0.99	0.04	0.00	0.04	30.12	36.24
GLDSU69	-282.000	-280.023	-280	0.99	0.02	0.00	0.02	16.41	17.40
GLDSU70	-286.000	-280.049	-280	0.99	0.03	0.00	0.03	29.52	21.30
GLDSU72	-294.000	-290.009	-290	1.00	0.02	0.00	0.02	18.23	17.43
GLDSU73	-298.000	-290.046	-290	0.99	0.02	0.00	0.02	16.86	20.41
GLDSU74	-302.000	-300.047	-300	0.98	0.03	0.01	0.04	10.52	10.53
GLDSU75	-306.000	-300.048	-300	0.99	0.02	0.00	0.02	13.06	20.34
GLDSU77	-313.000	-310.049	-310	0.98	0.04	0.00	0.04	6.73	8.90
GLDSU78	-317.000	-310.047	-310	0.99	0.03	0.00	0.03	7.81	7.20
GLDSU79	-321.000	-320.049	-320	0.95	0.03	0.00	0.03	2.74	3.57
GLDSU80	-325.000	-320.047	-320	0.99	0.04	0.00	0.04	4.94	6.53
GLDSU81	-329.000	-320.049	-320	0.99	0.03	0.00	0.03	4.22	5.34
GLDSU82	-333.000	-330.049	-330	0.98	0.05	0.00	0.05	2.70	2.47
GLDSU83	-337.000	-330.050	-330	0.99	0.03	0.00	0.03	2.61	3.39
GLDSU84	-341.000	-340.047	-340	0.95	0.03	0.00	0.03	0.86	1.00
GLDSU85	-344.000	-340.047	-340	0.99	0.02	0.00	0.02	1.37	1.41
GLDSU86	-348.000	-340.046	-340	0.99	0.04	0.00	0.04	1.35	1.76
GLDSU87	-352.000	-350.046	-350	0.98	0.04	0.00	0.04	0.62	0.60
GLDSU88	-356.000	-350.050	-350	0.99	0.02	0.00	0.02	0.66	0.52
GLDSU90	-364.000	-360.046	-360	0.99	0.01	0.00	0.01	0.22	0.23
GLDSU91	-368.000	-360.046	-360	0.99	0.04	0.00	0.04	0.20	0.21
GLDSU92	-372.000	-370.045	-370	0.98	0.04	0.01	0.05	0.06	0.12
GLDSU93	-376.000	-370.048	-370	0.99	0.04	0.00	0.04	0.07	0.10
GLDSU94	-379.000	-370.049	-370	0.99	0.02	0.00	0.02	0.08	0.11
GLDSU95	-383.000	-380.048	-380	0.98	0.04	0.00	0.04	0.03	0.03
GLDSU96	-387.000	-380.046	-380	0.99	0.04	0.01	0.05	0.02	0.03
GLDSU97	-391.000	-390.049	-390	0.95	0.02	0.00	0.02	0.00	0.00
GLDSU98	-395.000	-390.048	-390	0.99	0.03	0.00	0.03	0.00	0.01
Average				0.97	0.03	0.00	0.03	8.36	8.80
Std. Dev.				0.13	0.01	0.01	0.01	10.77	10.93

Table 6: Results for solution of IPs for Set (A), class G-C(L)-S.

Instance	LP Value	SD Value (SDTR)	IP Value	Gap Red.	Run Time (s) SDTR	CPLEX with SD Cut	SDTR + CPLEX	CPLEX	Cplex with LP Cut
GCLS2	-452.000	-451.049	-451	0.95	0.03	0.00	0.03	0.00	0.00
GCLS5	-454.000	-453.042	-453	0.96	0.01	0.00	0.01	0.00	0.01
GCLS7	-456.000	-455.040	-455	0.96	0.01	0.00	0.01	0.01	0.02
GCLS10	-458.000	-457.038	-457	0.96	0.01	0.00	0.01	0.05	0.05
GCLS12	-460.000	-459.037	-459	0.96	0.01	0.00	0.01	0.17	0.18
GCLS13	-460.000	-459.037	-459	0.96	0.01	0.00	0.01	0.17	0.18
GCLS15	-462.000	-461.035	-461	0.96	0.01	0.00	0.01	0.41	0.45
GCLS17	-464.000	-463.034	-463	0.97	0.01	0.00	0.01	1.07	1.13
GCLS18	-464.000	-463.034	-463	0.97	0.01	0.00	0.01	1.08	1.12
GCLS20	-466.000	-465.032	-465	0.97	0.01	0.00	0.01	2.38	2.49
GCLS22	-468.000	-467.030	-467	0.97	0.00	0.00	0.00	4.08	4.49
GCLS23	-468.000	-467.030	-467	0.97	0.00	0.00	0.00	4.07	4.49
GCLS25	-470.000	-469.049	-469	0.95	0.00	0.00	0.00	7.67	8.04
GCLS28	-472.000	-471.048	-471	0.95	0.00	0.00	0.00	13.84	13.40
GCLS30	-474.000	-473.046	-473	0.95	0.01	0.00	0.01	22.53	22.66
GCLS33	-476.000	-475.044	-475	0.96	0.01	0.00	0.01	32.17	34.18
GCLS35	-478.000	-477.043	-477	0.96	0.00	0.00	0.00	45.21	43.21
GCLS38	-480.000	-479.041	-479	0.96	0.00	0.00	0.00	57.32	58.90
GCLS40	-482.000	-481.040	-481	0.96	0.01	0.00	0.01	72.46	74.63
GCLS41	-482.000	-481.040	-481	0.96	0.01	0.00	0.01	72.76	72.65
GCLS43	-484.000	-483.038	-483	0.96	0.00	0.00	0.00	87.77	88.68
GCLS45	-486.000	-485.036	-485	0.96	0.00	0.00	0.00	100.31	99.14
GCLS46	-486.000	-485.036	-485	0.96	0.00	0.00	0.00	99.92	99.60
GCLS48	-488.000	-487.035	-487	0.96	0.00	0.00	0.00	107.06	110.46
GCLS50	-490.000	-489.033	-489	0.97	0.00	0.00	0.00	112.34	117.52
GCLS51	-490.000	-489.033	-489	0.97	0.00	0.00	0.00	112.00	117.80
GCLS53	-492.000	-491.031	-491	0.97	0.00	0.00	0.00	110.07	121.70
GCLS56	-494.000	-493.030	-493	0.97	0.00	0.00	0.00	105.66	107.24
GCLS58	-496.000	-495.028	-495	0.97	0.00	0.00	0.00	97.04	96.67
GCLS61	-498.000	-497.027	-497	0.97	0.00	0.00	0.00	84.03	90.39
GCLS63	-500.000	-499.010	-499	0.99	0.00	0.00	0.00	71.42	72.10
GCLS64	-500.000	-499.010	-499	0.99	0.00	0.00	0.00	71.66	71.27
GCLS66	-502.000	-501.048	-501	0.95	0.00	0.00	0.00	57.44	57.14
GCLS68	-504.000	-503.046	-503	0.95	0.01	0.00	0.01	43.39	43.66
GCLS69	-504.000	-503.046	-503	0.95	0.00	0.00	0.00	43.11	43.07
GCLS71	-506.000	-505.045	-505	0.95	0.01	0.00	0.01	30.86	30.26
GCLS73	-508.000	-507.043	-507	0.96	0.00	0.00	0.00	20.98	20.94
GCLS74	-508.000	-507.043	-507	0.96	0.01	0.00	0.01	20.98	20.58
GCLS76	-510.000	-509.041	-509	0.96	0.00	0.01	0.01	13.05	12.87
GCLS79	-512.000	-511.039	-511	0.96	0.00	0.00	0.00	7.66	8.11
GCLS81	-514.000	-513.037	-513	0.96	0.00	0.00	0.00	4.35	4.50
GCLS84	-516.000	-515.036	-515	0.96	0.00	0.00	0.00	2.25	2.38
GCLS86	-518.000	-517.034	-517	0.97	0.01	0.00	0.01	1.04	1.17
GCLS89	-520.000	-519.032	-519	0.97	0.00	0.00	0.00	0.47	0.52
GCLS91	-522.000	-521.030	-521	0.97	0.00	0.01	0.01	0.19	0.00
GCLS92	-522.000	-521.030	-521	0.97	0.01	0.00	0.01	0.20	0.00
GCLS94	-524.000	-523.028	-523	0.97	0.01	0.00	0.01	0.07	0.08
GCLS96	-526.000	-525.027	-525	0.97	0.01	0.00	0.01	0.02	0.02
GCLS97	-526.000	-525.027	-525	0.97	0.01	0.00	0.01	0.02	0.02
GCLS99	-528.000	-527.025	-527	0.98	0.00	0.00	0.00	0.00	0.00
Average				0.96	0.00	0.00	0.01	34.86	35.60
Std. Dev.				0.01	0.01	0.00	0.01	40.11	41.34

Table 7: Results for solution of IPs for Set (A), class G-C(L)-D(S).

Instance	LP Value	SD Value (SDTR)	IP Value	Gap Red.	Run Time (s)				
					SDTR	CPLEX with SD Cut	SDTR + CPLEX	CPLEX	Cplex with LP Cut
GCLDS7	-454.000	-453.042	-453	0.96	0.01	0.00	0.01	0.00	0.00
GCLDS8	-454.000	-453.042	-453	0.96	0.01	0.00	0.01	0.01	0.00
GCLDS10	-456.000	-455.040	-455	0.96	0.00	0.00	0.00	0.00	0.01
GCLDS12	-458.000	-457.038	-457	0.96	0.01	0.00	0.01	0.03	0.03
GCLDS13	-458.000	-457.038	-457	0.96	0.01	0.00	0.01	0.07	0.08
GCLDS15	-460.000	-459.037	-459	0.96	0.00	0.01	0.01	0.05	0.08
GCLDS17	-462.000	-461.035	-461	0.96	0.01	0.00	0.01	0.18	0.20
GCLDS18	-462.000	-461.035	-461	0.96	0.00	0.01	0.01	0.41	0.44
GCLDS20	-464.000	-463.034	-463	0.97	0.01	0.00	0.01	0.46	0.50
GCLDS23	-466.000	-465.032	-465	0.97	0.00	0.01	0.01	2.37	2.50
GCLDS25	-468.000	-467.030	-467	0.97	0.00	0.00	0.00	2.05	2.39
GCLDS28	-470.000	-469.049	-469	0.95	0.00	0.00	0.00	7.63	8.16
GCLDS30	-472.000	-471.048	-471	0.95	0.01	0.00	0.01	7.29	7.59
GCLDS33	-474.000	-473.046	-473	0.95	0.00	0.00	0.00	13.67	14.42
GCLDS35	-476.000	-475.044	-475	0.96	0.00	0.00	0.00	21.24	22.10
GCLDS36	-476.000	-475.044	-475	0.96	0.00	0.00	0.00	30.65	34.24
GCLDS38	-478.000	-477.043	-477	0.96	0.01	0.00	0.01	29.60	30.14
GCLDS40	-480.000	-479.041	-479	0.96	0.00	0.00	0.00	40.09	39.87
GCLDS41	-480.000	-479.041	-479	0.96	0.00	0.00	0.00	56.53	56.93
GCLDS43	-482.000	-481.040	-481	0.96	0.00	0.00	0.00	51.16	52.38
GCLDS46	-484.000	-483.038	-483	0.96	0.00	0.00	0.00	87.94	87.83
GCLDS48	-486.000	-485.036	-485	0.96	0.00	0.00	0.00	83.22	76.73
GCLDS51	-488.000	-487.035	-487	0.96	0.00	0.00	0.00	111.47	107.77
GCLDS53	-490.000	-489.033	-489	0.97	0.01	0.00	0.01	90.24	92.90
GCLDS56	-492.000	-491.031	-491	0.97	0.00	0.00	0.00	94.52	100.58
GCLDS58	-494.000	-493.030	-493	0.97	0.00	0.00	0.00	88.40	88.48
GCLDS59	-494.000	-493.030	-493	0.97	0.00	0.00	0.00	104.55	107.02
GCLDS61	-496.000	-495.028	-495	0.97	0.00	0.00	0.00	86.69	81.10
GCLDS63	-498.000	-497.027	-497	0.97	0.00	0.00	0.00	78.79	75.56
GCLDS64	-498.000	-497.027	-497	0.97	0.00	0.00	0.00	83.78	90.07
GCLDS66	-500.000	-499.010	-499	0.99	0.00	0.00	0.00	62.33	64.50
GCLDS68	-502.000	-501.048	-501	0.95	0.00	0.00	0.00	50.30	50.78
GCLDS69	-502.000	-501.048	-501	0.95	0.00	0.00	0.00	55.73	57.02
GCLDS71	-504.000	-503.046	-503	0.95	0.00	0.00	0.00	38.66	42.04
GCLDS74	-506.000	-505.045	-505	0.95	0.01	0.00	0.01	29.64	30.58
GCLDS76	-508.000	-507.043	-507	0.96	0.00	0.00	0.00	19.59	19.99
GCLDS79	-510.000	-509.041	-509	0.96	0.01	0.00	0.01	12.88	13.20
GCLDS81	-512.000	-511.039	-511	0.96	0.01	0.00	0.01	7.55	7.90
GCLDS82	-512.000	-511.039	-511	0.96	0.00	0.00	0.00	7.66	8.00
GCLDS84	-514.000	-513.037	-513	0.96	0.00	0.00	0.00	3.92	4.43
GCLDS86	-516.000	-515.036	-515	0.96	0.00	0.00	0.00	2.05	2.40
GCLDS87	-516.000	-515.036	-515	0.96	0.01	0.00	0.01	2.23	2.34
GCLDS89	-518.000	-517.034	-517	0.97	0.00	0.00	0.00	1.04	1.15
GCLDS91	-520.000	-519.032	-519	0.97	0.00	0.00	0.00	0.39	0.53
GCLDS92	-520.000	-519.032	-519	0.97	0.00	0.00	0.00	0.47	0.51
GCLDS94	-522.000	-521.030	-521	0.97	0.00	0.00	0.00	0.19	0.22
GCLDS97	-524.000	-523.028	-523	0.97	0.01	0.00	0.01	0.07	0.07
Average				0.96	0.00	0.00	0.00	31.23	31.61
Std. Dev.				0.01	0.00	0.00	0.00	35.85	35.90

Since we do not delete any subgradients in an unbroken inner loop $\underline{V}_{k,l}(u_k) \geq \underline{V}_k(u_k) = V_{\alpha_k}(u_k)$ and as $\underline{V}_{k,l}$ is always a minorant of V_{α_k} we have

$$V_{\alpha_k}(u_k) \geq \underline{V}_{k,l}(u_k) \geq V_{\alpha_k}(u_k) \tag{19}$$

giving equality and hence

$$\underline{V}_{k,l}(u_k) - \underline{V}_{k,l}(u_{k,l}) \geq V_{\alpha_k}(u_k) - v_k^*.$$

Thus (14) holds.

When $\tau = 1$ is infeasible then $\tau = \Delta_{k+l} / \left\|u^k - p_{\alpha_k}(u_k)\right\|_\infty$ is certainly feasible for (18) thus

$$\begin{aligned}
\underline{V}_{k,l}(u_{k,l}) &\leq \underline{V}_{k,l}\left(u_k + \frac{\Delta_{k,l}}{\|u^k - p_{\alpha_k}(u_k)\|_\infty}\left(p_{\alpha_k}(u_k) - u_k\right)\right) \\
&\leq V_{\alpha_k}\left(u_k + \frac{\Delta_{k,l}}{\|u^k - p_{\alpha_k}(u_k)\|_\infty}\left(p_{\alpha_k}(u_k) - u_k\right)\right) \\
\text{(by convexity)} \quad &\leq V_{\alpha_k}(u_k) + \frac{\Delta_{k,l}}{\|u^k - p_{\alpha_k}(u_k)\|_\infty}\left(V_{\alpha_k}(p_{\alpha_k}(u_k)) - V_{\alpha_k}(u_k)\right) \\
&= V_{\alpha_k}(u_k) + \frac{\Delta_{k,l}}{\|u^k - p_{\alpha_k}(u_k)\|_\infty}\left(v_k^* - V_{\alpha_k}(u_k)\right).
\end{aligned}$$

As $Ax_k - b$ is a subgradient of $V_{\alpha_k}(\cdot)$ at u_k we have $V_{\alpha_k}(u_k) = \underline{V}_{k,l}(u_k)$ and so

$$\underline{V}_{k,l}(u_k) - \underline{V}_{k,l}(u_{k,l}) \geq \frac{\Delta_{k,l}}{\|u^k - p_{\alpha_k}(u_k)\|_\infty}\left[V_{\alpha_k}(u_k) - v_k^*\right].$$

Thus (14) holds in this case. ■

The following is the proof of Lemma 6.

Proof. Suppose that we get an infinite sequence of trust region iterates that fail the termination test and hence $\rho_{k,l} \leq \xi$ for $l \geq l_1$ and also fail to satisfy (15), that is

$$V_{\alpha_k}(u_k) - \underline{V}_{k,q}(u_{k,q}) > \eta\left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_1}(u_{k,l_1})\right], \quad \text{for all } q \geq l_1. \tag{20}$$

As we do not delete sub-gradients during such trust region iterates we have for all $q > l \geq l_1$ that

$$\underline{V}_{k,q}(u_{k,l}) = V_{\alpha_k}(u_{k,l}). \tag{21}$$

As $\underline{V}_{k,q}(\cdot)$ is always a minorant of $V_{\alpha_k}(\cdot)$ and (21) holds we immediately have $\partial\underline{V}_{k,q}(u_{k+l}) \subseteq \partial V_{\alpha_k}(u_{k,l})$. As $s = Ax - b \in \partial V_{\alpha_k}(u_{k,l})$ for $V_{\alpha_k}(u_{k,l}) \geq 0$ and $\alpha_k \leq \bar{\alpha}$ implies

$$x \in \{x \in X \mid cx \leq \bar{\alpha}\} = X(\bar{\alpha})$$

which is by assumption bounded we have for any $\alpha_k \leq \bar{\alpha}$ that

$$\sup\{\|s\| \mid s \in \partial V_{\alpha_k}(u)\} \leq M < +\infty. \tag{22}$$

As $\rho_{k,l} \leq \xi$ for $l \geq l_1$ we have

$$\begin{aligned}
\underline{V}_{k,q}(u_{k,l}) = V_{\alpha_k}(u_{k,l}) &\geq V_{\alpha_k}(u_k) - \xi\left[V_{\alpha_k}(u_k) - \underline{V}_{k,l}(u_{k,l})\right] \\
&\geq V_{\alpha_k}(u_k) - \xi\left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_1}(u_{k,l_1})\right]
\end{aligned}$$

using the fact that the minorizing estimate grows monotonically. Thus

$$V_{\alpha_k}(u_k) - \underline{V}_{k,q}(u_{k,l}) \leq \xi\left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_1}(u_{k,l_1})\right]. \tag{23}$$

As $\eta \in (\xi, 1)$ we may define a neighbourhood of u_{k+l} as follows:

$$\|u - u_{k,l}\|_\infty \leq \frac{\eta - \xi}{M}\left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_1}(u_{k,l_1})\right] := \zeta > 0. \tag{24}$$

Using the definition of sub-gradients and the bound (22) we have for all $s \in \partial \underline{V}_{k,q}\ (u_{k,l})$

$$\begin{aligned}\underline{V}_{k,q}(u_{k,l}) - \underline{V}_{k,q}(u) &\leq s(u_{k,l} - u) \\ &\leq M \|u_{k,l} - u\|_\infty \leq (\eta - \xi)\left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_1}(u_{k,l_1})\right].\end{aligned}$$

Combining this with (23) we have

$$\begin{aligned}V_{\alpha_k}(u_k) - \underline{V}_{k,q}(u) &= \left[V_{\alpha_k}(u_k) - \underline{V}_{k,q}(u_{k,l})\right] + \left[\underline{V}_{k,q}(u_{k,l}) - \underline{V}_{k,q}(u)\right] \\ &\leq \eta\left[V_{\alpha_k}(u_k) - \underline{V}_{k,l_1}(u_{k,l_1})\right].\end{aligned}$$

It follows from (20), (24) and the previous bound that $u_{k,q}$, the solution to the trust region problem at iteration $k+q$ cannot be in the neighbourhood of radius ζ around $u_{k,l}$ i.e.

$$\|u_{k,q} - u_{k,l}\| > \zeta.$$

As we are in an infinite sequence of iterates for which $\rho_{k,l} \leq \xi$ for $q > l \geq l_1$ we have $\Delta_{k,l} \geq \|u_{k,l} - u_k\|_\infty \to 0$ but by definition of feasibility for the $k+q$th trust region problem.,

$$\|u_{k,q} - u_k\| \leq \Delta_{k,q} \to 0$$

leading to a contradiction. ■

References

[1] Karwan M. and Radin R. (1984), *Surrogate Dual Multiplier Search Procedures in Integer Programming,* Operations Research, 32, No. 1, pp. 52-69.

[2] Karwan M. and Radin R. (1979), *Some Relationships between Lagrangian and Surrogate Duality in Integer Programming*, Mathematical Programming 17, 320-334.

[3] Kim S.-L. and Kim S. (1998), *Exact Algorithm for the Surrogate Dual of an Integer Programming Problem: Subgradient Method Approach*, JOTA, 96, No. 2, pp. 363-375.

[4] Li D. and Sun X. (2006), *Nonlinear Integer Programming*, International Series in Operations Resarch & Management, Springer.

[5] Linderoth J. and Wright S. (2003), *Decomposition Algorithms for Stochastic Programming on a Computational Grid,* Computational Optimization and Applications, Kluwer Acad. Publ., 24, pp 207-250.

[6] Noll D. (2012), *Bundle method for non-convex minimization with inexact subgradients and function values*, Computational and Analytical Mathematics, Springer Proceedings in Mathematics, `http://www.math.univ-toulouse.fr/~noll/preprints.html`.

[7] R. T. Rockafellar and R. J.-B. Wets (1998), *Variational Analysis*, A series of Comprehensive Studies in Mathematics, Pub. Springer.

[8] Sarin S., Karwan M. and Radin R. (1988), *Surrogate duality in a branch-and-bound procedure for integer programming*, European Journal of Operational Research, 33, pp. 326-333.

[9] Han, B. and Leblet, J. and Simon, G., *Hard multidimensional multiple choice knapsack problems, an empirical study*, Computers and Operations Research, 37, pp 172–181.

[10] http://www-user.tu-chemnitz.de/ helmberg/ConicBundle/

[11] ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/MMKP/