Nondifferentiable Optimization of Lagrangian Dual Formulations for Linear Programs with Recovery of Primal Solutions

Churlzu Lim

Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial and Systems Engineering

Hanif D. Sherali, Chair
Ebru K. Bish
Barbara M. P. Fraticelli
G. V. Loganathan
Subhash C. Sarin

May 28, 2004
Blacksburg, Virginia

Keywords: Nondifferentiable Optimization (NDO), Nonsmooth Optimization, Mixed-Integer Programming (MIP), Lagrangian Relaxation, Variable Target Value Method, Primal Recovery.

Nondifferentiable Optimization of Lagrangian Dual Formulations for Linear Programs with Recovery of Primal Solutions

Churlzu Lim

(ABSTRACT)

This dissertation is concerned with solving large-scale, ill-structured linear programming (LP) problems via Lagrangian dual (LD) reformulations. A principal motivation for this work arises in the context of solving mixed-integer programming (MIP) problems where LP relaxations, sometimes in higher dimensional spaces, are widely used for bounding and cut-generation purposes. Often, such relaxations turn out to be large-sized, ill-conditioned problems for which simplex as well as interior point based methods can tend to be ineffective. In contrast, Lagrangian relaxation or dual formulations, when applied in concert with suitable primal recovery strategies, have the potential for providing quick bounds as well as enabling useful branching mechanisms. However, the objective function of the Lagrangian dual is nondifferentiable, and hence, we cannot apply popular gradient or Hessian-based optimization techniques that are commonly used in differentiable optimization. Moreover, the subgradient methods that are popularly used are typically slow to converge and tend to stall while yet remote from optimality. On the other hand, more advanced methods, such as the bundle method and the space dilation method, involve additional computational and storage requirements that make them impractical for large-scale applications. Furthermore, although we might derive an optimal or near-optimal solution for LD, depending on the dual-adequacy of the methodology used, a primal solution may not be available. While some algorithmically simple primal solution recovery schemes have been developed in theory to accompany Lagrangian dual optimization, their practical performance has been disappointing. Rectifying these inadequacies is a challenging task that constitutes the focal point for this dissertation. Many practical applications dealing with production planning and control,

engineering design, and decision-making in different operational settings fall within the purview of this context and stand to gain by advances in this technology.

With this motivation, our primary interests in the present research effort are to develop effective nondifferentiable optimization (NDO) methods for solving Lagrangian duals of large-sized linear programs, and to design practical primal solution recovery techniques. This contribution would then facilitate the derivation of quick bounds/cuts and branching mechanisms in the context of branch-and-bound/cut methodologies for solving mixed-integer programming problems.

We begin our research by adapting the Volume Algorithm (VA) of Barahona and Anbil (2000) developed at IBM as a direction-finding strategy within the variable target value method (VTVM) of Sherali et al. (2000). This adaptation makes VA resemble a deflected subgradient scheme in contrast with the bundle type interpretation afforded by the modification of VA as proposed by Bahiense et al. (2002). Although VA was originally developed in order to recover a primal optimal solution, we first present an example to demonstrate that it might indeed converge to a nonoptimal primal solution. However, under a suitable condition on the geometric moving average factor, we establish the convergence of the proposed algorithm in the dual space. A detailed computational study reveals that this approach yields a competitive procedure as compared with alternative strategies including the average direction strategy (ADS) of Sherali and Ulular (1989), a modified Polyak-Kelley cutting-plane strategy (PKC) designed by Sherali et al. (2001), and the modified Volume Algorithm routines RVA and BVA proposed by Bahiense et al. (2002), all embedded within the same VTVM framework. As far as CPU times are concerned, the VA strategy consumed the least computational effort for most problems to attain a near-optimal objective value. Moreover, the VA, ADS, and PKC strategies revealed considerable savings in CPU effort over a popular commercial linear program solver, CPLEX Version 8.1, when used to derive near-optimal solutions.

Next, we consider two variable target value methods, the Level Algorithm of Brännlund (1993) and VTVM, which require no prior knowledge of upper bounds on the

optimal objective value while guaranteeing convergence to an optimal solution. We evaluate these two algorithms in conjunction with various direction-finding and step-length strategies such as PS, ADS, VA, and PKC. Furthermore, we generalize the PKC strategy by further modifying the cut's right-hand-side values and additionally performing sequential projections onto some previously generated Polyak-Kelley's cutting-planes. We call this a generalized PKC (GPKC) strategy. Moreover, we point out some latent deficiencies in the two aforementioned variable target value algorithms in regard to their target value update mechanisms, and we suggest modifications in order to alleviate these shortcomings. We further explore an additional local search procedure to strengthen the performance of the algorithms. Noting that no related convergence analyses have been presented, we prove the convergence of the Level Algorithm when used in conjunction with the ADS, VA, or GPKC schemes. We also establish the convergence of VTVM when employing GPKC. Based on our computational study, the modified VTVM algorithm produced the best quality solutions when implemented with the GPKC strategy, where the latter performs sequential projections onto the four most recently generated Polyak-Kelley cutting-planes as available. Also, we demonstrate that the proposed modifications and the local search technique significantly improve the overall performance. Moreover, the VTVM procedure was observed to consistently outperform the Level Algorithm as well as a popular heuristic subgradient method of Held et al. (1974) that is widely used in practice. As far as CPU times are concerned, the modified VTVM procedure in concert with the GPKC strategy revealed the best performance, providing near-optimal solutions in about 27.84% of the effort at an average as that required by CPLEX 8.1 to produce the same quality solutions.

We next consider the Lagrangian dual of a bounded-variable equality constrained linear programming problem. We develop two novel approaches for solving this problem, which attempt to circumvent or obviate the nondifferentiability of the objective function. First, noting that the Lagrangian dual function is differentiable almost everywhere, whenever the NDO algorithm encounters a nondifferentiable point, we employ a proposed *perturbation technique* (PT) in order to detect a differentiable point in the vicinity of the current solution from which a further search can be conducted. In a second approach, called

the *barrier-Lagrangian dual reformulation* (BLR) method, the primal problem is reformulated by constructing a barrier function for the set of bounding constraints such that an optimal solution to the original problem can be recovered by suitably adjusting the barrier parameter. However, instead of solving the barrier problem itself, we dualize the equality constraints to formulate a Lagrangian dual function, which is shown to be twice differentiable. Since differentiable pathways are made available via these two proposed techniques, we can advantageously utilize differentiable optimization methods along with popular conjugate gradient schemes. Based on these constructs, we propose an algorithmic procedure that consists of two sequential phases. In Phase I, the PT and BLR methods along with various deflected gradient strategies are utilized, and then, in Phase II, we switch to the modified VTVM algorithm in concert with GPKC (VTVM-GPKC) that revealed the best performance in the previous study. We also designed two target value initialization methods to commence Phase II, based on the output from Phase I. The computational results reveal that Phase I indeed helps to significantly improve the algorithmic performance as compared with implementing VTVM-GPKC alone, even though the latter was run for twice as many iterations as used in the two-phase procedures. Among the implemented procedures, the PT method in concert with certain prescribed deflection and Phase II initialization schemes yielded the best overall quality solutions and CPU time performance, consuming only 3.19% of the effort as that required by CPLEX 8.1 to produce comparable solutions. Moreover, we also tested some ergodic primal recovery strategies with and without employing BLR as a warm-start, and observed that an initial BLR phase can significantly enhance the convergence of such primal recovery schemes.

Having observed that the VTVM algorithm requires the fine-tuning of several parameters for different classes of problems in order to improve its performance, our next research investigation focuses on developing a robust variable target value framework that entails the management of only a few parameters. We therefore design a novel algorithm, called the *Trust Region Target Value* (TRTV) method, in which a trust region is constructed in the dual space, and its center and size are adjusted in a manner that eventually induces a dual optimum to lie at the center of the hypercube trust region. A

related convergence analysis has also been conducted for this procedure. We additionally examined a variation of TRTV, where the hyperrectangular trust region is more effectively adjusted for normalizing the effects of the dual variables. In our computational study, we compared the performance of TRTV with that of the VTVM-GPKC procedure. For four direction-finding strategies (PS, VA, ADS, and GPKC), the TRTV algorithm consistently produced better quality solutions than did VTVM-GPKC. The best performance was obtained when TRTV was employed in concert with the PS strategy. Moreover, we observed that the solution quality produced by TRTV was consistently better than that obtained via VTVM, hence lending a greater degree of robustness. As far as computational effort is concerned, the TRTV-PS combination consumed only 4.94% of the CPU time required by CPLEX 8.1 at an average in order to find comparable quality solutions. Therefore, based on our extensive set of test problems, it appears that the TRTV along with the PS strategy is the best and the most robust procedure among those tested.

Finally, we explore an outer-linearization (or cutting-plane) method along with a trust region approach for refining available dual solutions and recovering a primal optimum in the process. This method enables us to escape from a jamming phenomenon experienced at a non-optimal point, which commonly occurs when applying NDO methods, as well as to refine the available dual solution toward a dual optimum. Furthermore, we can recover a primal optimal solution when the resulting dual solution is close enough to a dual optimum, without generating a potentially excessive set of constraints. In our computational study, we tested two such trust region strategies, the Box-step (BS) method of Marsten et al. (1975) and a new Box Trust Region (BTR) approach, both appended to the foregoing TRTV-PS dual optimization methodology. Furthermore, we also experimented with deleting nonbinding constraints when the number of cuts exceeds a prescribed threshold value. This proposed refinement was able to further improve the solution quality, reducing the near-zero relative optimality gap for TRTV-PS by 20.6-32.8%. The best strategy turned out to be using the BTR method while deleting nonbinding constraints (BTR-D). As far as the recovery of optimal solutions is concerned, the BTR-D scheme resulted in the best measure of primal feasibility, and

although it was terminated after it had accumulated only 50 constraints, it revealed a better performance than the ergodic primal recovery scheme of Shor (1985) that was run for 2000 iterations while also assuming knowledge of the optimal objective value in the dual scheme.

In closing, we mention that there exist many optimization methods for complex systems such as communication network design, semiconductor manufacturing, and supply chain management, that have been formulated as large-sized mixed-integer programs, but for which deriving even near-optimal solutions has been elusive due to their exorbitant computational requirements. Noting that the computational effort for solving mixed-integer programs via branch-and-bound/cut methods strongly depends on the effectiveness with which the underlying linear programming relaxations can be solved, applying theoretically convergent and practically effective NDO methods in concert with efficient primal recovery procedures to suitable Lagrangian dual reformulations of these relaxations can significantly enhance the overall computational performance of these methods. We therefore hope that the methodologies designed and analyzed in this research effort will have a notable positive impact on analyzing such complex systems.

Acknowledgments

First of all, I would like to give my best thanks to Dr. Hanif Sherali for his paradigmatic supervision through my Ph.D. studies. I am so much obliged to him for his persistent help and leadership. I am also beholden to my committee members, Dr. Ebru Bish, Dr. Barbara Fraticelli, Dr. G.V. Loganathan, and Dr. Subhash Sarin, for their warmhearted guidance and encouragement. I thank my parents, Wan Chul Lim and Hwa Ja Song, and my parents-in-law, Eui-Jun Kim and Mae-Ja Park, who have given support and encouragement throughout my Ph.D. education. Finally, I would like to acknowledge my wife, Dr. Hye-Young Kim, and lovely daughter, Harin Lim, for their consistent love and support.

Table of Contents

List of Figures

List of Tables

Chapter 1:
Introduction

1.1 Problem Description

Consider the following nondifferentiable optimization (NDO) problem:

$$\textbf{NDO:} \quad \begin{array}{ll} \text{Maximize} & \theta(\pi) \\ \text{subject to} & \pi \in \Pi, \end{array} \tag{1.1}$$

where $\Pi \subset R^m$ is a convex set, and where $\theta : R^m \rightarrow R^1$ is a concave nondifferentiable function. NDO problems appear in various applications such as maximizing the minimum of functions (e.g., in Lagrangian duality or game theory problems), employing exact penalty functions for solving constrained nonlinear optimization problems, and in certain optimal control problems (see Shor (1985) and Goffin and Vial (2002), for example).

While the methods developed in this research effort can be equally applied to any NDO problem of the form in (1.1), we particularly focus on an important special case that arises in the context of solving linear programming (LP) relaxations of mixed-integer programs (MIPs) via Lagrangian dual formulations. LP relaxations, sometimes in higher dimensional spaces, are the most widely used bounding and cut-generation mechanisms for solving MIP problems. For the most part, simplex-based and interior point approaches are employed for solving such relaxations, largely because of readily available software. Despite the stability and robustness of these software packages, both interior point and simplex algorithms can stall for large ill-conditioned problems of the type that result when applying several of the reformulation techniques that are commonly used for solving discrete problems (see Sherali and Myers (1988), Sherali and Ulular (1989), Adams and Sherali (1993), Sherali and Tuncbilek (1997), and Barahona and Anbil (2000)). Besides, such large-scale relaxations can impose high storage requirements. However, near-optimal solutions that can be obtained by solving suitable Lagrangian dual formulations of these LP relaxations, can reasonably serve the purpose of obtaining lower

bounds and cuts in the context of branch-and-bound/cut methods. In addition, if a near-optimal corresponding primal solution can be derived, this could be helpful in selecting branching variables, finding improved incumbent solutions, and applying reliable termination rules for the dual ascent scheme based on the duality gap.

To be more specific, let us consider the following general 0-1 mixed-integer program.

MIP: Minimize $c^T x$

subject to $Ax \geq b$

$$x \in X_I \equiv \left\{ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in R^n : Dx \geq d, 0 \leq x_1 \leq u, \ x_2 \text{ binary} \right\},$$

where $x_1 \in R^{n_1}$, $x_2 \in R^{n_2}$, $n = n_1 + n_2$, and where A is $m \times n$ and D is $l \times n$. The dimensions of $b, c, u,$ and d are in accordance with those of $x_1, x_2, A,$ and D, and a superscript T will throughout denote the transpose operation. Also, all vectors are assumed to be column vectors. The linear programming relaxation of MIP is given by

LR: Minimize $c^T x$

subject to $Ax \geq b$ (1.2)

$$x \in X \equiv \left\{ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in R^n : Dx \geq d, 0 \leq x_1 \leq u, \ 0 \leq x_2 \leq e \right\},$$

where e will always refer to an appropriately dimensioned vector of ones.

The constraint set $Dx \geq d$ is assumed to be specially structured so that it permits an easy solution of the Lagrangian dual subproblem given below, while retaining sufficient control to facilitate the derivation of relatively sharp bounds within a reasonable effort. When choosing such constraints, there always exists a trade-off between the ease in solving the underlying subproblems and the sharpness of the bounds derived therefrom, both in theory where discrete relationships might be retained in the subproblems, as well as more generally in practice (see Fisher (1981), Sherali and Myers

(1988), and Sherali and Ulular (1989)). The constraint $x_1 \le u$, possibly redundant, is added to X in order to ensure dual feasibility. By dualizing the set of constraints $Ax \ge b$, we obtain the following Lagrangian dual problem for LR.

LD: Maximize $\theta(\pi)$ (1.3a)

subject to $\pi \ge 0$, (1.3b)

where $\theta(\pi) = \min\{c^T x + \pi^T (b - Ax) : x \in X\}$. (1.3c)

Note that $\pi \in R^m$ is the dual variable vector associated with the set of constraints $Ax \ge b$. The problem (1.3c) that evaluates the *Lagrangian dual function* θ is referred to as the *Lagrangian dual subproblem*.

When we solve LD, two major questions arise. First, the Lagrangian dual function is not differentiable because the solution set of the subproblem may not be a singleton (see Ch. 6, Bazaraa et al. (1993)). Therefore, we cannot apply popular gradient or Hessian-based optimization techniques that are commonly used in differentiable optimization. Secondly, although we might derive an optimal or near-optimal solution of LD, depending on the dual-adequacy of the methodology used, a primal solution may not be available.

1.2 Literature Review

Several approaches have been developed for solving NDO problems, and they all have a similar basic skeletal structure. Their main point of difference lies in the specification of search directions and the selection of step-lengths. To present the schema of this underlying algorithmic process, consider the NDO problem in the form of (1.1). The general structure of algorithms for such nondifferentiable optimization can then be described as follows.

Initialization Step: Choose a starting point $\pi^1 \in \Pi$. Let the current incumbent solution and objective value be $\bar{\pi} = \pi^1$ and $\bar{\theta} = \theta(\pi^1)$, respectively. Set the iteration counter to $k = 1$, and go to the main step.

Main Step: Given π^k, find a subgradient $g^k \in \partial\theta(\pi^k)$, where $\partial\theta(\pi^k)$ is the subdifferential of θ at π^k. If $g^k = 0$, then stop with the incumbent solution $\bar{\pi}$ as optimal. Otherwise, find a search direction d^k using the subgradient g^k along with some suitable previously generated information, select an appropriate step-length $\lambda_k > 0$, and compute $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$, where $P_\Pi(\cdot)$ is a projection onto Π and $\hat{\pi}^{k+1} = \pi^k + \lambda_k d^k$. If $\theta_{k+1} \equiv \theta(\pi^{k+1}) > \bar{\theta}$, put $\bar{\pi} = \pi^{k+1}$ and $\bar{\theta} = \theta(\pi^{k+1})$. Increase k by 1, and repeat the main step.

The desirable choices of the direction, d^k, and the step-length, λ_k, have been the principal points of research focus in solving nondifferentiable optimization problems. Among various alternatives for selecting search directions, the *pure subgradient* algorithm has been the most widely used method for nondifferentiable optimization. This algorithm, which was pioneered by Polyak (1967, 1969) and popularized by Held et al. (1974), imitates the steepest ascent method of differentiable optimization by replacing the gradient by the subgradient, prescribing the direction $d^k = g^k$.

In spite of its simplicity, the pure subgradient algorithm has the weakness of a slow convergence rate due to the choice of a direction that is prone to cause zigzagging. In addition, such a direction (as with most NDO methods) might not even ensure an ascent, a phenomenon that is further aggravated by an inadequate control on the step-length (see Shor (1985)). Moreover, this approach does not admit a satisfactory practical termination rule other than a maximum limit on the number of iterations. Because of these reasons, there have been many attempts to find improved directions accompanied by suitable step-length choices.

Polyak (1967) has shown that the convergence of the pure subgradient algorithm is guaranteed if the step-length is chosen to satisfy $\lambda_k \to 0$ and $\sum \lambda_k \to \infty$ as $k \to \infty$. However, it is difficult to choose a step-length that would avoid slow convergence while satisfying these conditions. Polyak (1969) also devised an alternative step-length as

$$\lambda_k = \frac{\beta[\theta^* - \theta(\pi^k)]}{\left\| g^k \right\|^2}, \tag{1.4}$$

where θ^* is the optimal objective value and where $\varepsilon_1 \le \beta \le 2 - \varepsilon_2$ for some ε_1, $\varepsilon_2 > 0$. This choice of step-length assures a geometric convergence rate under some additional assumptions on θ. (See Konnov (2003) for an extension of these convergence properties for normalized subgradient algorithms as applied to NDO problems having a quasiconcave nondifferentiable objective function.) In most cases, however, the optimal objective value, θ^*, is obviously unknown. Held et al. (1974) devised a practical variation of this step-length rule by replacing θ^* with a fixed upper bound on the optimal objective value, and by periodically halving the value of β, with an initial stage-length of $2m$, until some threshold value is reached. Bazaraa and Sherali (1981) proposed using a convex combination of a fixed upper bound and the incumbent objective value as an estimate for θ^*, where the convex combination weight was updated periodically. This scheme was further explored and modified by Fumero (2001). Sherali and Myers (1988) tested various alternatives for selecting step-lengths and found that the combination in which Held et al.'s method was used first, and then a switchover was made to the Bazaraa and Sherali technique after some iterations, achieved relatively good results. Sen and Sherali (1986), and Sherali and Ulular (1989) used a step-length proportional to the difference between the current penalty function value and the Lagrangian function value in the context of a primal-dual conjugate subgradient algorithm.

Kim et al. (1991) proposed using an estimate for the optimal objective value, called a *target value*, and adjusting this value in a suitable manner so that the resulting algorithm converges to an optimum under certain additional assumptions regarding the availability of upper bounds on the optimal objective value, and on the Euclidean distances between an optimum and the iterates generated by the algorithm. Brännlund (1993) and Sherali et al. (2000) developed variable target value schemes, the *Level Algorithm* and the *Variable Target Value Method* (VTVM), respectively, which make no prior assumption on the availability of any bounds while guaranteeing convergence to an optimum. Goffin and Kiwiel (1999) have also provided a convergence analysis for a

simple variant of the Level Algorithm.

As mentioned earlier, since the pure subgradient algorithm imitates the steepest ascent scheme of differentiable optimization, it shares the same deficiency of being prone to a zigzagging behavior. Shor (1985) describes this behavior by pointing out that the directions tend to asymptotically become nearly orthogonal to the direction toward the set of optimal solutions. Striking an analogy with conjugate gradient methods for the differentiable case, conjugate or deflected subgradient algorithms were therefore introduced to overcome this slow convergence rate, by taking a combination of the previous direction and the current subgradient to derive a new direction. Camerini et al. (1975) used such a deflection method in their modified gradient technique. Sherali and Ulular (1989) developed the *average direction strategy* (ADS) in which the adopted direction bisects the angle between the current subgradient and the previous direction. Brännlund (1993) deflected subgradients based on level sets whenever a slow convergence or a zigzagging phenomenon was detected. In the context of Lagrangian dual optimization, Barahona and Anbil (2000) took a geometric moving average of current and past solutions of the Lagrangian subproblems, instead of using only the current solution, when calculating directions in their so-called *Volume Algorithm* (VA).

Mifflin (1977) and Lemarechal (1977) proposed a different class of approach for NDO problems, called a *bundle method*, in which a new direction is obtained from the set of previous subgradients, as opposed to using only the current iteration information (see also Kiwiel (1991) and Brännlund et al. (1995)). Although bundle methods guarantee a monotone increase in the objective function value, the main difficulty of this method is that it requires a quadratic subproblem to be solved at each iteration. For large-sized problems, this can become unreasonably expensive. (See Makela (2002), for a recent survey of bundle methods.)

Another approach that also accommodates previous iteration information is the space dilation method, which is the nondifferentiable analog of quasi-Newton methods. Similar to the use of the Hessian matrix or its approximation in quasi-Newton approaches, a suitable matrix is generated to premultiply the subgradient in order to deflect the direction toward an optimal direction (see Shor (1970a, 1970b)). Shor and Zhurbenko (1971) and Shor and Shabashova (1972) performed space dilation along the

difference of two consecutive subgradients in their popular *r*-algorithm. Although the *r*-algorithm showed good computational performance, it required the storage and inversion of a matrix. This detracts from the simplicity of subgradient methods and can become impractical for large-sized problems. To alleviate this difficulty, Sherali et al. (2001) proposed memoryless and limited memory updates. They also proposed a combination of their variable target value method with a *Polyak-Kelley cutting-plane* (PKC) strategy in order to construct an algorithm that was demonstrated to be a significant improvement over the *r*-algorithm.

Besides optimizing the nondifferentiable Lagrangian function when solving Lagrangian duals of linear programs, the recovery of primal solutions is another important consideration. Having a good quality primal solution for LP relaxations can serve not only to fathom nodes and to select branching variables in the context of branch-and-bound methods for MIPs, but it can also facilitate a reliable and practical termination criterion based on the duality gap. However, in general, a subgradient approach does not provide primal solutions. Sen and Sherali (1986) and Sherali and Ulular (1989) have developed primal-dual subgradient algorithms that simultaneously coordinate the solution of a suitable penalty function and a Lagrangian dual problem. Although such primal-dual subgradient techniques can provide primal and dual solutions, they impose the extra burden of designing suitable primal penalty function parameter update schemes, and require the optimization to be conducted jointly in both the primal and dual spaces.

A computationally simpler idea for designing primal recovery methods for Lagrangian dual optimization is to construct a sequence of ergodic primal iterates during the normal course of the algorithm that would converge to a dual solution. This sequence can be composed by using the solutions generated for subproblems up to the current iteration as shown by Shor (1985), and Larsson and Liu (1989), who proved the convergence of such sequences when using specific weighting schemes. Sherali and Choi (1996) extended such choices of weights and proved convergence when employing more popular and effective step-length choices for both pure and deflected subgradient algorithms. Larsson et al. (1999) proposed some further choices of weights that yield primal convergence for more general convex programs. Barahona and Anbil (2000) showed that the solution to a Dantzig-Wolfe decomposition-based reformulation of LR

(that is equivalent to a suitable Lagrangian dual (LD) formulation) could be expressed as the ratio of a specified volume below each active face to the total volume of a particular bounded polyhedron that is defined by the dualized objective function and the level set in the vicinity of an optimal solution to LD. The prescribed solution to the underlying Dantzig-Wolfe decomposition of LR was accordingly estimated via a geometric moving average of solutions to the Lagrangian subproblem (1.3c) in their heuristic method, Volume Algorithm.

1.3 Proposed Research and Organization of this Dissertation

Our research effort commences with embedding the Volume Algorithm (VA) of Barahona and Anbil (2000) as a direction-finding strategy within the VTVM algorithm of Sherali et al. (2000). In their Volume Algorithm, Barahona and Anbil employed a geometric moving average of current and past subgradients in an attempt to generate improving directions. Noting that no convergence analysis was provided by Barahona and Anbil, Bahiense et al. (2002) interpreted VA as a variant of the bundle method, and proposed modified routines RVA and BVA that were proven to converge in the dual space under restricted conditions regarding the sequence of iterates generated. Moreover, these methods require suitable upper bounds on the optimal objective value.

On the other hand, we adapt the VA strategy as a deflected subgradient scheme within the VTVM framework, and provide a convergence analysis under a suitable simple condition on the geometric moving average factor. This adaptation affords implementing VA without the need for devising specialized problem-specific heuristics for computing upper bounds as in Bahiense et al. (2002). Using a variety of test problems, we compare the performance of the proposed algorithm against RVA and BVA strategies within the VTVM algorithm. We also present the performance of the average direction strategy (ADS) of Sherali and Ulular (1989) and the Polyak-Kelley cutting-plane method (PKC) implemented by Sherali et al. (2001), which revealed a promising performance in the respective computational studies. Based on our computational experience, the PKC strategy affords the best performance in terms of the quality and robustness of solutions produced, with the VA and ADS schemes coming in a relatively close second and third, respectively. Concerning CPU times consumed by each

algorithm, the VA strategy yields the least computational effort, while PKC is the second best.

Note that the design of the Volume Algorithm was originally motivated to recover a primal optimum by estimating the solution to the underlying Dantzig-Wolfe decomposition of the primal problem with a geometric moving average of the solutions to the corresponding Lagrangian subproblems. However, we present a counter-example to exhibit that the primal convergence is indeed not guaranteed via the generic Volume Algorithm.

Next, we consider two variable target value methods, the Level Algorithm and VTVM, which require no prior knowledge of upper bounds on the optimal objective value while guaranteeing convergence to an optimal solution. We evaluate these two algorithms in conjunction with various direction-finding and step-length strategies such as the pure subgradient method (PS), ADS, VA, and PKC. Furthermore, we generalize the PKC strategy by further modifying the cut's right-hand-side values and additionally performing sequential projections onto some previously generated Kelley's cutting-planes. We call this generalized PKC approach GPKC. Moreover, we point out some latent deficiencies in the two aforementioned variable target value algorithms in regard to their target value update mechanisms, and we suggest modifications in order to alleviate these shortcomings. We further explore an additional local search procedure to strengthen the performance of the algorithms. Computational results are reported to compare performances of two variable target value methods, and to demonstrate the efficacy of proposed modifications and local search.

The convergence for VTVM when used in conjunction with a deflected subgradient strategy such as ADS was provided by Sherali et al. (2000). Likewise, we establish the convergence of VTVM when employing the GPKC strategy. Furthermore, Goffin and Kiwiel (1999) have provided a convergence analysis for the Level Algorithm when implemented in conjunction with the PS strategy. However, no convergence analysis for deflected subgradient methods was presented. We therefore establish the convergence of the Level Algorithm when used in conjunction with the ADS, VA, or GPKC schemes.

We next consider the Lagrangian dual of a bounded-variable equality constrained

linear programming problem. We develop two novel approaches for solving this problem, which attempt to circumvent or obviate the nondifferentiability of the objective function. First, noting that the Lagrangian dual function is differentiable almost everywhere, whenever the NDO algorithm encounters a nondifferentiable point, we exploit perturbations in order to search for a differentiable point in the vicinity of the point via our *perturbation technique* (PT). In a second approach, called the *barrier-Lagrangian dual reformulation* (BLR) method, the primal problem is reformulated by constructing barrier function for the set of bounding constraints. Note that an optimal solution to the original problem can be recovered by suitably adjusting the barrier parameter. However, instead of solving the barrier problem itself, we solve its Lagrangian dual, which yields a differentiable objective function. Consequently, we can apply conventional differentiable optimization methods to solve this problem, including several deflected gradient strategies along with a quadratic-fit line-search procedure. Moreover, this methodology can produce near-optimal primal solutions as well. We utilize this approach as an initialization phase for the VTVM algorithm in the hope of enhancing its computational performance by providing a relatively good dual starting solution along with a judicious initial target value. Related computational results display the efficacy of the proposed methods.

We observed from our preliminary numerical study that the VTVM algorithm can experience an adverse crawling phenomenon for some classes of problems, and also, it requires the fine-tuning of several parameter values for different classes of problems. Motivated with this non-robustness of VTVM, we focus next on developing a new variable target value method, a so-called *Trust Region Target Value* (TRTV) method, which entails the management of only a few parameters. In this algorithm, we manipulate a hyperrectangular trust region in the dual space so that an optimal dual solution is eventually recovered as its central point. We also provide the convergence analysis of TRTV when used in conjunction with a deflected subgradient strategy. Furthermore, we suggest using a more generalized hyperrectangular trust region that considers the effect of normalizing the primal constraints. Computational results are reported in order to compare the performances of TRTV and VTVM, and these results amply demonstrate the relative robustness of TRTV.

Finally, we turn our attention to designing two dual solution refinement techniques together with a practical primal recovery procedure. Given a near dual optimal solution provided by an NDO algorithm, we utilize an outer-linearization technique to further refine the dual solution as well as recover a near-optimal primal optimal solution. However, in order to avoid the generation of a potentially excessive set of constraints in the master problem that arises in the context of the outer-linearization algorithm, we utilize two trust region approaches: the *Box-step method* of Marsten et al. (1975) and a *Box Trust Region* imposed in the dual space. Moreover, whenever the NDO algorithm stalls at a non-optimal dual solution, the proposed refinement techniques can be invoked as a side-step routine to help escape from the current jammed non-optimal solution. Relevant computational results are provided to demonstrate the usefulness of the proposed refinement and primal recovery schemes.

The remainder of this dissertation is organized as follows. In Chapter 2, we develop a detailed procedure for adapting the VA strategy within the VTVM algorithm. We provide its convergence analysis and compare its computational performance against RVA, BVA, ADS, and PKC within the VTVM framework. We also present a counter-example to exhibit that the generic Volume Algorithm of Barahona and Anbil (2000) might not recover a primal optimum. Chapter 3 studies two variable target value frameworks, the Level Algorithm and VTVM, along with various direction-finding and step-length strategies such as PS, ADS, VA, and GPKC. We provide convergence proofs for various combinations of the two variable target value frameworks used in concert with different direction-finding and step-length strategies for which no prior convergence analysis existed. We also recommend some computational expedients for the Level Algorithm and the VTVM procedure.

In Chapter 4, we present the aforementioned perturbation technique and the barrier-Lagrangian reformulation approach, both as stand-alone routines as well as in a two-phase algorithmic procedure wherein these procedures are used to initialize the VTVM-based approach. Chapter 5 provides a motivation for and a detailed derivation of the TRTV method along with its convergence analysis and some proposed variations. Chapter 6 provides two dual solution refinement and primal recovery procedures via an outer-linearization algorithm using a Box-step or a Box Trust Region approach. Finally, a

summary of our research contributions and recommendations for future research are presented in Chapter 7.

Chapter 2:
Embedding the Volume Algorithm in a Variable Target Value Method

2.1 Introduction

Consider the problem LD stated in (1.3). As mentioned earlier, although the step-length choice according to (1.4) that is employed in the popular subgradient-based methods for solving NDO problems guarantees convergence, the optimal objective value, θ^*, is unavailable *a priori* in most cases. Hence, on the one hand, a variety of practical heuristic methods have been developed that replace θ^* with some upper bound on θ and adjust either this bound or the factor β in (1.4) periodically in attempting to obtain good computational performance (see Held et al. (1974), Bazaraa and Sherali (1981), and Fumero (2001)). On the other hand, the Level Algorithm (Brännlund (1993), Goffin and Kiwiel (1999)), the Variable Target Value Subgradient Method (Kim et al. (1991)), and the Variable Target Value Method (VTVM, Sherali et al. (2000)) manage a target value that replaces θ^* in (1.4), guaranteeing convergence to an optimal solution under various stated conditions (see review given in Chapter 1).

As far as the direction-finding strategy is concerned, Barahona and Anbil (2000) employed a fixed geometric moving average of current and past subgradients, instead of solving quadratic subproblems as in bundle methods, to obtain such a weighting scheme for their so-called Volume Algorithm (VA). (See Bahiense et al. (2002) for an interpretation of VA as a bundle type method.) However, Barahona and Anbil did not provide any convergence analysis. Bahiense et al. (2002) modified this procedure to propose a *Revised Volume Algorithm* (RVA), which has a restricted convergence property under the assumption that the sequences $\{\pi^k\}$ and $\{g^k\}$ are convergent when finite steps that incur sufficient improvements from previous outer-loop iterates (or *stability centers*) are generated by the algorithm. They also developed an alternative bundle method variant, called BVA, which belongs to the class of *penalized bundle methods* (see Hiriart-

Urruty and Lemarechal, 1993). Note that all these methods: VA, RVA, and BVA, require some scheme for deriving upper bounds on θ to be practically implementable. For the Rectlinear Steiner problem used in their computational experiment, Bahiense et al. implemented three particular heuristics to compute upper bounds. However, a general guideline to obtain these upper bounds has not been provided for either RVA or BVA.

In contrast, we employ the VTVM algorithm of Sherali et al. (2000) in this chapter for maintaining a target value via a simple update scheme that guarantees convergence while not requiring the computation of any such upper bounds on the problem. For a direction strategy, we exploit the geometric moving average concept of the generic Volume Algorithm, but use it similar to a conjugate subgradient (or deflected subgradient) type approach, rather than as a bundle method. Hence, we show that the Volume Algorithm can be embedded within the framework of a variable target value method as a conjugate subgradient strategy in order to derive an effective, provably convergent algorithm.

The remainder of this chapter is organized as follows. In the next section, we briefly review the generic Volume Algorithm and provide an example to illustrate that this procedure might not recover a primal optimum. In Section 2.3, we describe the details of the proposed algorithm, and then establish its convergence in Section 2.4. In Section 2.5, we present computational results using four types of test problems that are either randomly generated or are retrieved from the OR Library (Beasley, 1990). We also compare the performance of employing VA as a deflection strategy against RVA and BVA of Bahiense et al. (2002) within the VTVM framework as summarized in Section 2.3. Furthermore, we also present the performance of an alternative deflection scheme, namely, the average direction strategy (ADS) of Sherali and Ulular (1989) and the Polyak-Kelley cutting-plane method (PKC) of Sherali et al. (2001), which have been shown to yield a promising computational performance in their respective numerical studies. Finally, some concluding remarks are given in Section 2.6.

2.2 Volume Algorithm

A general statement of the Volume Algorithm for solving LD, which is pertinent to the analysis of the present chapter, is given below.

Volume Algorithm (VA)

STEP 0: Select an initial solution π^1, and compute $\theta_1 \equiv \theta(\pi^1)$ via (1.3c) along with a corresponding optimum x^1 to this Lagrangian subproblem. Let $\bar{\pi}^1 = \pi^1$. Choose a moving average parameter $\alpha \in [0,1]$ and a step-length parameter $\beta \in (0,2)$. Put $\bar{x}^1 = x^1$, and initialize the iteration counters $k = 1$ and $l = 1$.

STEP 1: Select the search direction $d^k = b - A\bar{x}^k$, and compute a step-length $\lambda_k = \beta \dfrac{UB - \theta(\bar{\pi}^l)}{\left\| d^k \right\|^2}$, where UB is an upper bound on the optimal value for LD.

STEP 2: Set $\pi^{k+1} = P_\Pi(\bar{\pi}^l + \lambda_k d^k)$, where $P_\Pi(\cdot)$ is a projection onto Π, and compute $\theta_{k+1} \equiv \theta(\pi^{k+1})$ along with its solution x^{k+1}. If $\theta_{k+1} > \theta(\bar{\pi}^l)$ (called *serious-step*), then set $\bar{\pi}^{l+1} = \pi^{k+1}$, and increment $l \leftarrow l+1$. Update $\bar{x}^{k+1} = \alpha x^{k+1} + (1-\alpha)\bar{x}^k$, and increment $k \leftarrow k+1$. Return to Step 1.

This method was originally motivated by the desire to simultaneously recover a primal optimal solution via the moving average sequence $\{\bar{x}^k\}$. Considering the dual to the master program of the Danzig-Wolfe decomposition method applied to LR, Barahona and Anbil first showed that the optimal solution to this master program can be represented as the ratio of the volume associated with each active dual constraint at an optimum to the total volume of the region in the $\{(\pi, \theta) \in R^{m+1}\}$-space encompassed by the active dual constraints at the optimum and the plane $\theta \geq \theta^* - \varepsilon$, where $\varepsilon > 0$. Using this observation, Barahona and Anbil proposed a practical implementation scheme in which, at iteration k, the foregoing ratio was approximated via the geometric moving average coefficients $\alpha, \alpha(1-\alpha), \alpha(1-\alpha)^2, \ldots, \alpha(1-\alpha)^k$, intending that $\bar{x}^k$ would then approach a primal optimal solution in the limit. However, we present a counter example below to show that a primal optimum may not be recovered by using the above Volume Algorithm.

Consider the following simple one dimensional linear program.

$$\text{Minimize} \quad x \tag{2.1a}$$

$$\text{subject to} \quad x \geq 1 \tag{2.1b}$$

$$0 \leq x \leq 2\,. \tag{2.1c}$$

Dualizing the constraint $x \geq 1$ in (2.1b), we obtain the corresponding Lagrangian dual problem as:

$$\text{Maximize} \quad \theta(\pi) = \min\{x + \pi(1-x) : 0 \leq x \leq 2\} \tag{2.2a}$$

$$\text{subject to} \quad \pi \geq 0\,. \tag{2.2b}$$

Figure 2.1 illustrates the Lagrangian dual function. Suppose that we apply the Volume Algorithm using the parameter values $\alpha = 0.5$ and $\beta = 0.5$. Furthermore, assume that the initial dual solution is chosen as $\pi^1 = 0$, and UB is chosen as the optimal objective value 1. We wish to demonstrate that the Volume Algorithm would then produce a sequence $\{\bar{x}^k\}$ such that $\bar{x}^k = 0$, $\forall k$, which is not the primal optimal solution (indeed, it is infeasible). Notice that $x^k = 0$ uniquely evaluates $\theta(\pi^k)$ in (2.2a) whenever $0 \leq \pi^k < 1$. Hence, it is sufficient to show that the Volume Algorithm produces a sequence of dual iterates $\pi^k \in [0,1)$, $\forall k$. We exhibit this by induction. For $k = 1$, we have $\pi^1 = 0 \in [0,1)$.

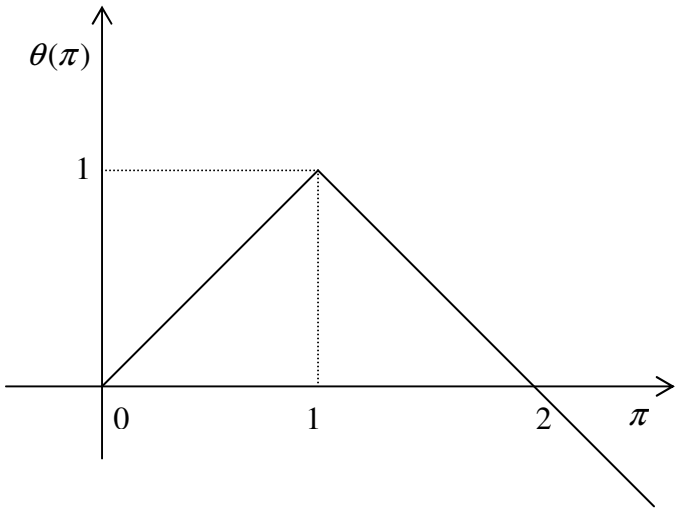

Figure 2.1 Lagrangian Dual Function for the Counter Example.

Now, for any $k \geq 2$, assume that $\pi^1, \ldots, \pi^{k-1}$ are in the interval $[0,1)$. Denoting a subgradient of θ at π^k by g^k, we have, from Steps 1 and 2 of the algorithm,

$$\begin{aligned} d^{k-1} &= b - A\bar{x}^{k-1} = b - A[\alpha x^{k-1} + (1-\alpha)\bar{x}^{k-2}] \\ &= \alpha(b - Ax^{k-1}) + (1-\alpha)(b - A\bar{x}^{k-2}) = \alpha g^{k-1} + (1-\alpha)d^{k-2} \\ &= 0.5g^{k-1} + 0.5d^{k-2} = 0.5g^{k-1} + 0.5(0.5g^{k-2} + 0.5d^{k-3}) = \cdots = 1, \end{aligned}$$

because $g^i = 1$ for $i = 1, \ldots, k-1$ and $d^1 = g^1$. Notice that we have $\bar{\pi}^i = \pi^i$ for $i = 1, \ldots, k-1$, since every step induces an improvement with $\theta_i = \pi^i$, $i = 1, \ldots, k-1$. In fact, this improving sequence of iterates is given via

$$\pi^i = \pi^{i-1} + \beta \frac{UB - \theta_{i-1}}{\left\| d^{i-1} \right\|^2} d^{i-1} = \pi^{i-1} + 0.5(1 - \pi^{i-1}) = 0.5 + 0.5\pi^{i-1}, \quad i = 2, \ldots, k-1.$$

Furthermore, by the same schema, using $i = k$ above, we get $\pi^k = 0.5 + 0.5\pi^{k-1}$. Note that $0.5\pi^{k-1} < 0.5$ since $\pi^{k-1} < 1$. Therefore, we have $\pi^k \in [0,1)$ as well, thereby leading to nonconvergence. Although the Volume Algorithm may not recover a primal solution, it can still produce dual iterates that converge to a dual optimum. Observe that in the above example, we get that $\lim_{k \to \infty} \pi^k = 0.5 + 0.5^2 + 0.5^3 + \cdots = 1$, so that the dual scheme indeed converges to the optimum.

2.3 VTVM along with the VA Strategy for Search Directions

In this section, we outline the VTVM algorithm that employs the Volume Algorithm as a deflected subgradient direction-finding strategy. The problem to be solved is the Lagrangian dual problem as given by (1.3). In the VTVM algorithm of Sherali et al. (2000), the target value is revised based on the recent history of iterates, where it is increased following an acceptable level of improvement, or it is decreased following a specified number of iterations without attaining the designated improvement. In this

process, the acceptable improvement tolerance parameter is set as some fraction of the difference between the revised target value and the incumbent objective value, or some threshold value $\varepsilon > 0$, whichever is greater. As usual, let us denote the LD objective value $\theta(\pi^k)$ by θ_k, and a subgradient of $\theta(\cdot)$ at π^k by g^k, for any iterate $\pi^k \in \Pi$. Note that such a subgradient can be computed as $g^k = b - Ax^k$, where x^k is a solution to the Lagrangian subproblem (1.3c) that evaluates $\theta(\pi^k)$. The details of the general VTVM algorithm can be summarized as follows, using the notation and parameter specifications given below.

ε_0 ≡ termination tolerance on subgradient norms

ε ≡ threshold value for designating an acceptable improvement

β ≡ initial step-length parameter

σ ≡ factor for updating the target value

k ≡ overall iteration counter

$k_{\max}$ ≡ maximum number of iterations

τ ≡ iteration counter while yet attempting to attain the targeted acceptable level of improvement

$\bar{\tau}$ ≡ maximum number of iterations tolerated without attaining an acceptable level of improvement

γ ≡ iteration counter for consecutive non-improvements

$\bar{\gamma}$ ≡ maximum number of consecutive non-improving iterations permitted before resetting and updating parameter

l ≡ target value adjustment (or outer iteration) counter

ε_l ≡ acceptable improvement tolerance parameter

η ≡ algorithmic parameter related to increasing the target value

Δ ≡ partial cumulative amount of improvement

VA within the VTVM Algorithm

Initialization Step: Select $\varepsilon_0 \geq 0$, $\varepsilon > 0$, $k_{\max}$, $\beta \in (0,1]$, $\sigma \in (0,1/3]$, $\bar{\tau}$, $\bar{\gamma}$, and $\eta \in (0,1]$. Choose an initial solution $\pi^1 \in \Pi$, and determine θ_1 and g^1. Set the incumbent vectors $\bar{\pi} = \pi^1$, $\bar{g} = g^1$, and the incumbent objective value $z_1 = \theta_1$. If $\left\| g^1 \right\| \leq \varepsilon_0$, terminate the algorithm. Initialize $l = 1$, $k = 1$, $\tau = 0$, $\gamma = 0$, $\Delta = 0$, and the reset indicator RESET=1. Let the target value be initialized as $w_1 = \min\left\{ UB, \theta_1 + \left\| g^1 \right\|^2 / 2 \right\}$ where *UB* is some known upper bound on the problem, if available (else, $UB = \infty$). Set $\varepsilon_1 = \sigma(w_1 - \theta_1)$.

STEP 1: Call Subroutine VA (specified below) to obtain π^{k+1}. Increment $\tau \leftarrow \tau + 1$ and $k \leftarrow k + 1$. Compute θ_k and g^k. Put RESET=0.

STEP 2: If $\theta_k > z_{k-1}$, set $\Delta \leftarrow \Delta + (\theta_k - z_{k-1})$, $z_k = \theta_k$, $\bar{\pi} = \pi^k$, $\bar{g} = g^k$, $\gamma = 0$, and proceed to Step 3. Otherwise, put $z_k = z_{k-1}$, increment $\gamma \leftarrow \gamma + 1$, and go to Step 4.

STEP 3: If $k > k_{\max}$ or $\left\| g^k \right\| \leq \varepsilon_0$, terminate the algorithm. If $z_k \geq w_l - \varepsilon_l$, go to Step 5. If $\tau \geq \bar{\tau}$, go to Step 6. Else, return to Step 1.

STEP 4: If $k > k_{\max}$, terminate the algorithm. If $\gamma \geq \bar{\gamma}$ or $\tau \geq \bar{\tau}$, go to Step 6. Else, return to Step 1.

STEP 5: Compute $w_{l+1} = z_k + \varepsilon_l + \eta\Delta$. Let $\varepsilon_{l+1} = \max\{(w_{l+1} - z_k)\sigma, \varepsilon\}$, put $\tau = 0$, $\Delta = 0$, and increment $l \leftarrow l + 1$. Return to Step 1.

STEP 6: Compute $w_{l+1} = \dfrac{(z_k + \varepsilon_l) + w_l}{2}$, and $\varepsilon_{l+1} = \max\{(w_{l+1} - z_k)\sigma, \varepsilon\}$. If $\gamma \geq \bar{\gamma}$, then set $\bar{\gamma} \leftarrow \min\{\bar{\gamma} + 10, 50\}$. If $(w_l - w_{l+1}) \leq 0.1$, then set $\beta \leftarrow \max\{\beta/2,\ 10^{-6}\}$. Put $\gamma = 0$, $\tau = 0$, $\Delta = 0$, and $l \leftarrow l + 1$. Reset $\pi^k = \bar{\pi}$, $\theta_k = z_k$, $g^k = \bar{g}$, and put RESET=1. Return to Step 1.

Remark 2.1: For a practical implementation of the VTVM algorithm that yet guarantees convergence (see Sherali et al. (2000)), initializing $r = 0.1$ and setting $\bar{r} = r + 1$, we

update the target value when it is increased at Step 5 as $w_{l+1} = z_k + \max\{\varepsilon_l + \eta\Delta, r|z_k|\}$, where r is updated by $r \leftarrow r/\bar{r}$ whenever the second term in the maximand is greater.

Remark 2.2: The various parameter values used are prescribed as follows. We set $\varepsilon_0 = 10^{-6}$ and $k_{\max} = 2000$ for terminating the algorithm, and we select $\beta = 0.8$ for the initial step-length parameter. Furthermore, we select $\varepsilon = 0.1$, $\sigma = 0.15$, $\eta = 0.75$, $r = 0.1$ (see Remark 2.1), $\bar{\tau} = 75$, and $\bar{\gamma} = 20$ as recommended in Sherali et al. (2000). The initial solution is chosen as $\pi^1 = 0$.

Subroutine VA

STEP (i): If RESET=1, set the search direction as $d^k = g^k$. Otherwise, determine

$$d^k = \alpha g^k + (1-\alpha)d^{k-1}.$$

If $\|d^k\| = 0$ (practically, if $\|d^k\| \le \delta$ for some zero-tolerance $\delta > 0$), put $d^k = g^k$. Compute the step-length $\lambda_k = \beta[w_l - \theta_k] \big/ \|d^k\|^2$.

STEP (ii): Return $\pi^{k+1} = P_\Pi(\pi^k + \lambda_k d^k)$.

Remark 2.3: In addition to the VTVM parameters, we select $\alpha \in [\beta, 1]$ for the VA subroutine. (In our computational study, we set $\alpha = \beta = 0.8$.) Note that the original Volume Algorithm induces a new iterate from the incumbent solution $\bar{\pi}$, i.e., $\pi^{k+1} = P_\Pi(\bar{\pi} + \lambda_k d^k)$, whereas we adopt the VA strategy as a subgradient deflection scheme. In this case, as detailed above, given a current iterate π^k, the next iterate π^{k+1} is computed as $\pi^{k+1} = P_\Pi(\pi^k + \lambda_k d^k)$, where $d^k = \alpha g^k + (1-\alpha)d^{k-1}$. Moreover, in lieu of an upper bound *UB* on LD, we use the target value w_l instead for computing step-lengths.

For comparison purposes, we also implemented RVA and BVA of Bahiense et al. (2002) within the VTVM framework, where the target value of VTVM replaces upper bounds when calculating step-lengths. Here, the target value w and the step-length parameter β are adjusted as in VTVM, based on the progress of the iterates. For the sake of brevity, we suppress the repetition of this adjustment process, and focus on the RVA and BVA strategies in the statements given below.

RVA within the VTVM Framework

STEP 0: Select an initial solution π^1, compute $\theta_1 \equiv \theta(\pi^1)$ via (1.3c) along with a corresponding optimum x^1 to this Lagrangian subproblem, and put $g^1 = b - Ax^1$. Let $\bar{\pi}^1 = \pi^1$. Choose a step-length parameter $\beta \in (0,2)$. Put $\bar{x}^1 = x^1$, and initialize the iteration counters $k = 1$ and $l = 1$. Furthermore, select $m_1 \in (0,1)$, and initialize $\bar{\varepsilon}_1 = 0$, and $p^1 = \pi^1$.

STEP 1: Select the search direction $d^k = b - A\bar{x}^k$, and compute the step-length $\lambda_k = \beta[w - \theta(\bar{\pi}^l)] \Big/ \left\| d^k \right\|^2$, where w is the target value provided by VTVM.

STEP 2: Set $\pi^{k+1} = P_\Pi(\bar{\pi}^l + \lambda_k d^k)$, compute $\theta_{k+1} \equiv \theta(\pi^{k+1})$ along with its solution x^{k+1}, and put $g^{k+1} = b - Ax^{k+1}$. Determine the ascent measure $\delta_k = \lambda_k \left\| d^k \right\|^2 + \left| (d^k)^T (\bar{\pi}^l - p^k) \right| + \bar{\varepsilon}_k$. If $\theta_k \geq \theta(\bar{\pi}^l) + m_1 \delta_k$, then set $\bar{\pi}^{l+1} = \pi^{k+1}$, and increment $l \leftarrow l + 1$. Compute $E_k \equiv (g^{k+1})^T (\bar{\pi}^l - \pi^{k+1})$ and $\hat{E}_k \equiv (d^k)^T (\bar{\pi}^l - p^k) + \bar{\varepsilon}_k$. Let α_k be the solution to the problem:

$$\min_{\alpha \in [0,1]} \frac{\lambda_k}{2} \left\| \alpha g^{k+1} + (1-\alpha) d^k \right\|^2 + \alpha E_k + (1-\alpha) \hat{E}_k .$$

Update $\bar{x}^{k+1} = \alpha_k x^{k+1} + (1-\alpha_k)\bar{x}^k$, $p^{k+1} = \alpha_k \pi^{k+1} + (1-\alpha_k) p^k$, and $\bar{\varepsilon}_{k+1} = \alpha_k (1-\alpha_k)(g^{k+1} - d^k)^T (p^k - \pi^{k+1}) + (1-\alpha_k)\bar{\varepsilon}_k$. If the stopping rules of VTVM are satisfied, terminate the algorithm. Adjust the target value and the step-length parameter β, if necessary, by the VTVM algorithm. Increment

$k \leftarrow k+1$, and return to Step 1.

BVA within the VTVM Framework

STEP 0: Select an initial solution π^1, compute $\theta_1 \equiv \theta(\pi^1)$ via (1.3c) along with a corresponding optimum x^1 to this Lagrangian subproblem, and put $g^1 = b - Ax^1$. Let $\bar{\pi}^1 = \pi^1$. Choose a step-length parameter $\beta \in (0,2)$. Put $\bar{x}^1 = x^1$, and initialize the iteration counters $k = 1$ and $l = 1$. Furthermore, select $m_1 \in (0,1)$, and initialize $\bar{\varepsilon}_1 = 0$.

STEP 1: Select the search direction $d^k = b - A\bar{x}^k$, and compute the step-length $\lambda_k = \beta[w - \theta(\bar{\pi}^l)] \Big/ \left\| d^k \right\|^2$, where w is the target value provided by VTVM.

STEP 2: Set $\pi^{k+1} = P_{\Pi}(\bar{\pi}^l + \lambda_k d^k)$, compute $\theta_{k+1} \equiv \theta(\pi^{k+1})$ along with its solution x^{k+1}, and put $g^{k+1} = b - Ax^{k+1}$. Determine the ascent measure $\delta_k = \lambda_k \left\| d^k \right\|^2 + \bar{\varepsilon}_k$. If $\theta_k \geq \theta(\bar{\pi}^l) + m_1 \delta_k$, then set $\bar{\pi}^{l+1} = \pi^{k+1}$, and increment $l \leftarrow l+1$. Put $e_k = \theta_{k+1} + (g^{k+1})^T(\bar{\pi}^l - \pi^{k+1}) - \theta(\bar{\pi}^l)$. Let α_k be the solution to the problem:

$$\min_{\alpha \in [0,1]} \frac{\lambda_k}{2} \left\| \alpha g^{k+1} + (1-\alpha) d^k \right\|^2 + \alpha e_k + (1-\alpha)\bar{\varepsilon}_k .$$

Update $\bar{x}^{k+1} = \alpha_k x^{k+1} + (1-\alpha_k)\bar{x}^k$, and $\bar{\varepsilon}_{k+1} = \alpha_k e_k + (1-\alpha_k)\bar{\varepsilon}_k$. If the stopping rules of VTVM are satisfied, terminate the algorithm. Adjust the target value and the step-length parameter β, if necessary, by the VTVM algorithm. Increment $k \leftarrow k+1$, and return to Step 1.

Remark 2.4: As used in the numerical study of Bahiense et al. (2002), we set $m_1 = 0.001$. Note that, when computing the step-length, Barahona and Anbil used $\theta(\bar{\pi}^l)$ in their VA procedure, while Bahiense et al. used θ_k in their RVA and BVA methods. Because the next iterate is searched from the stability center, $\bar{\pi}^l$, it is reasonable to use

$\theta(\bar{\pi}^l)$ instead of θ_k, as done in Step 1 for all the above algorithms.

In addition, we compared the performance of the proposed algorithm against the average direction strategy (ADS) for subgradient deflection as suggested by Sherali and Ulular (1989), and the Polyak-Kelley cutting-plane (PKC) method of Sherali et al. (2001). The ADS strategy deflects the subgradient direction by bisecting the angle between it and the previous direction, and has been demonstrated to yield promising results among deflection schemes. On the other hand, Polyak (1969) implicitly determined the search direction and the step-length simultaneously by projecting the current solution onto a pair of Polyak-Kelley's cutting-planes. Sherali et al. (2001) replaced the unknown optimal objective value in the cutting-planes by the target value provided by VTVM, and this displayed the best performance among competing space dilation type strategies. The subroutines for the ADS and PKC, which replace Subroutine VA at Step 1 of the VTVM algorithm, are given below.

Subroutine ADS

STEP (i): If RESET=1, set the search direction as $d^k = g^k$. Otherwise, determine

$d^k = g^k + \dfrac{\left\|g^k\right\|}{\left\|d^{k-1}\right\|} d^{k-1}$. If $\left\|d^k\right\| = 0$ (practically, if $\left\|d^k\right\| \le \delta$ for some zero-tolerance $\delta > 0$), put $d^k = g^k$. Compute the step-length $\lambda_k = \beta[w_l - \theta_k] \Big/ \left\|d^k\right\|^2$.

STEP (ii): Return $\pi^{k+1} = P_\Pi(\pi^k + \lambda_k d^k)$.

Subroutine PKC

STEP (i): Set $\psi_1 = \dfrac{\beta[w_l - \theta_k]}{\left\|g^k\right\|^2}$, $\psi_2 = 0$, and $\hat{\pi}^{k+1} = \pi^k + \psi_1 g^k$. If RESET=1, return

$\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$.

STEP (ii): Compute $\xi_2 = \theta_k + \beta(w_l - \theta_k) - \theta_{k-1} + (\pi^{k-1})^T g^{k-1}$. If $\xi_2 \le (\hat{\pi}^{k+1})^T g^{k-1}$, return

$\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$.

STEP (iii): Compute $\xi_1 = \beta(w_l - \theta_k) + (\pi^k)^T g^k$. Put $\psi_1 = 0$, $\psi_2 = \dfrac{\xi_2 - (\pi^k)^T g^{k-1}}{\left\| g^{k-1} \right\|^2}$, and

let $\hat{\pi}^{k+1} = \pi^k + \psi_2 g^{k-1}$. If $\psi_2 > 0$ and $(\hat{\pi}^{k+1})^T g^k \geq \xi_1$, return $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$.

STEP (iv): Compute $\psi_0 = \left\| g^k \right\|^2 \left\| g^{k-1} \right\|^2 - \left((g^{k-1})^T g^k \right)^2$. If $\psi_0 < \delta$ for some zero-tolerance $\delta > 0$, put RESET=1 and return to Step (i). Otherwise, compute

$$\psi_1 = \left[\left\| g^{k-1} \right\|^2 \left(\xi_1 - (\pi^k)^T g^k \right) - \left((g^{k-1})^T g^k \right) \left(\xi_2 - (\pi^k)^T g^{k-1} \right) \right] \Big/ \psi_0 \text{, and}$$

$$\psi_2 = \left[\left\| g^k \right\|^2 \left(\xi_2 - (\pi^k)^T g^{k-1} \right) - \left((g^{k-1})^T g^k \right) \left(\xi_1 - (\pi^k)^T g^k \right) \right] \Big/ \psi_0 .$$

Put $\hat{\pi}^{k+1} = \pi^k + \psi_1 g^k + \psi_2 g^{k-1}$. Return $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$.

Remark 2.5: We used $\delta = 10^{-6}$ in Subroutine ADS as well as Subroutine PKC.

2.4 Convergence Analysis

In contrast with the case of differentiable optimization, a subgradient-based direction might not guarantee an ascent for a nondifferentiable objective function. Hence, standard convergence arguments based on monotone ascent steps along with closedness of algorithmic maps (see Bazaraa et al., 1993) are no longer applicable for nondifferentiable optimization. Instead, convergence proofs of algorithms for nondifferentiable optimization rely on the fact that a (suitably deflected) subgradient direction, when used in concert with a properly designed step-length strategy, can ensure a (periodically) monotone decrease in the distance toward an optimal solution, thereby guaranteeing the eventual convergence to optimality. In this section, we present a convergence analysis for the proposed algorithmic procedure. Note that if the algorithm terminates in a finite number of iterations with a zero subgradient, an optimal solution is at hand. Thus, in what follows, we ignore the termination rule predicated by $k_{\max}$, and additionally assume that the algorithm generates an infinite sequence $\{\pi^k\}$ using $\varepsilon_0 = 0$, i.e., we also suppose that the case of $\left\| g^k \right\| = 0$ for some k does not hold true.

The convergence analysis for the proposed algorithm is based in part on Sherali et

al.'s (2000) argument. Note that Lemma 2 and Lemma 3 in Sherali et al. (2000) do not depend on the specific search direction strategy used, and a key property to be satisfied for these results is that the sequence of distances between the iterates and an optimum, $\left\{\left\|\pi^k - \pi^*\right\|\right\}$, is a bounded monotone decreasing sequence, so that it is convergent. To show this, similar to Lemma 1 of Sherali et al. (2000), it is sufficient to prove that $(\pi^k - \pi^*)^T d^{k-1} \le 0$, as established in our Lemma 1 below.

Lemma 2.1

Consider the VA strategy, where the geometric moving average factor α and the initial step-length parameter β satisfy $0 < \beta \le \alpha \le 1$. Let us denote the target value at any iteration k by $\hat{w}_k$. Suppose that there exist values $\underline{w}$ and $\overline{w}$ such that

$$\theta_k \le \underline{w} < \hat{w}_k < \overline{w} \le \theta^*. \tag{2.3}$$

Then, the following inequality holds true:

$$(\pi^k - \pi^*)^T d^{k-1} \le 0, \quad \forall\, k, \tag{2.4}$$

where π^* is an optimal dual solution.

Proof: The proof is established by induction as follows. Noting that we start with $d^0 \equiv 0$, the base case for $k = 1$ is trivial. Now, assume that we have

$$(\pi^{k-1} - \pi^*)^T d^{k-2} \le 0 \quad \text{for any } k \ge 2,$$

and let us establish (2.4) at iteration k. Consider the following relation:

$$\begin{aligned}(\pi^k - \pi^*)^T d^{k-1} &= (\pi^k - \hat{\pi}^k + \hat{\pi}^k - \pi^*)^T d^{k-1} \\ &= (\pi^k - \hat{\pi}^k)^T d^{k-1} + (\hat{\pi}^k - \pi^*)^T d^{k-1}. \end{aligned} \tag{2.5}$$

Since Π is convex, π^{k-1} and $\pi^k \in \Pi$, and using $d^{k-1} = (1/\lambda_{k-1})(\hat{\pi}^k - \pi^{k-1})$, we get

$$(\pi^k - \hat{\pi}^k)^T d^{k-1} = (1/\lambda_{k-1})(\pi^k - \hat{\pi}^k)^T (\hat{\pi}^k - \pi^{k-1}) \le 0. \tag{2.6}$$

Furthermore, from the concavity of θ, the induction hypothesis, the inequalities in (2.3), and the condition that $0 < \beta \le \alpha \le 1$, we have

$$\begin{aligned}
(\hat{\pi}^k - \pi^*)^T d^{k-1} &= (\pi^{k-1} + \lambda_{k-1} d^{k-1} - \pi^*)^T d^{k-1} \\
&= (\pi^{k-1} - \pi^*)^T \left(\alpha g^{k-1} + (1-\alpha) d^{k-2}\right) + \beta(\hat{w}_{k-1} - \theta_{k-1}) \\
&\le -\alpha(\theta^* - \theta_{k-1}) + (1-\alpha)(\pi^{k-1} - \pi^*)^T d^{k-2} + \beta(\overline{w} - \theta_{k-1}) \\
&\le (\beta - \alpha)(\overline{w} - \theta_{k-1}) \le 0.
\end{aligned} \tag{2.7}$$

Using (2.6) and (2.7) within (2.5), we therefore have that (2.4) holds true. ■

The convergence of the VTVM algorithm along with the Subroutine VA then directly follows from Sherali et al. (2000) as follows.

Theorem 2.1 (Convergence of the VTVM Algorithm along with the Subroutine VA) Consider the VTVM algorithm along with the Subroutine VA, where the geometric moving average factor α and the initial step-length parameter β satisfy $0 < \beta \le \alpha \le 1$. Suppose that the sequences $\{\|\pi^k\|\}$ and $\{\|d^k\|\}$ generated by the algorithm are bounded. Then, for any $\varepsilon > 0$, the incumbent objective value sequence $\{z_k\}$ generated by the algorithm converges to $\bar{z}$, where $\bar{z} + \varepsilon \ge \theta^*$.

Proof: Using (2.4), it readily follows that $\{\|\pi^k - \pi^*\|\}$ is a bounded monotone decreasing sequence, and hence, Lemma 1 in Sherali et al. (2000) holds true. Furthermore, since

Lemma 2 and Lemma 3 in Sherali et al. (2000) hold true regardless of the direction-finding strategy, Theorem 1 in Sherali et al. (2000) holds true as well. ■

Remark 2.6: As noted in Sherali et al. (2000), the step-length parameter β is restricted to satisfy $0 < \beta \leq 1$ in order to establish the convergence of general conjugate subgradient strategies within the framework of VTVM, whereas convergence can be established when using pure subgradient (undeflected) directions by restricting β to lie in the interval $(0, 2)$. It is also interesting to note that we have an additional condition, $0 < \beta \leq \alpha \leq 1$, when using Subroutine VA in this context, in order to ensure convergence.

2.5 Computational Study

Three types of test problems - max cut, transportation, and general linear programming problems - are first used to compare the performance of the algorithm. The max cut problems are generated as in Barahona and Anbil (2000), while the sets of transportation and general linear programming problems are generated in the spirit of Rosen and Suzuki (1965), and Sherali et al. (2001). In addition, real-world set covering problems are taken from the OR-Library of the Imperial College Management School (Beasley, 1990), which lists a collection of test data sets for a variety of operations research problems. These set covering problems were formulated for crew scheduling in the Italian railways industry (Caprara et al., 1999).

Max Cut Problems

The max cut problem seeks a cut-set that maximizes the total weight on the edges connecting two subgraphs of a given complete graph having n vertices. The linear programming relaxation of a max cut problem having an objective weight vector c, which is formulated using the triangle-inequalities (2.8c) – (2.8e) to describe the cut-set, and where x_{ij} equals one if the edge (i, j) belongs to the cut-set and is zero otherwise, can be written as follows.

Table 2.1 Summary of Max Cut Test Problems.

Problem	Number of vertices	Number of variables	Number of constraints	CPLEX solution	
				Objective value	CPU time[1] (sec)
MC1	40	780	39520	-520	101.01
MC2	50	1225	78400	-816.67	368.65
MC3	60	1770	136880	-1180	1137.94
MC4	70	2415	218960	-1610	3561.68
MC5	80	3160	328640	-2106.67	62253.50

[1]CPLEX v. 8.1 is run on a Pentium III 667Mhz.

MC: Minimize $-c^T x$ (2.8a)

subject to $-x_{ij} - x_{jk} - x_{ik} \geq -2, \quad \forall\ 1 \leq i < j < k \leq n$ (2.8b)

$-x_{ij} + x_{jk} + x_{ik} \geq 0, \quad \forall\ 1 \leq i < j < k \leq n$ (2.8c)

$x_{ij} - x_{jk} + x_{ik} \geq 0, \quad \forall\ 1 \leq i < j < k \leq n$ (2.8d)

$x_{ij} + x_{jk} - x_{ik} \geq 0, \quad \forall\ 1 \leq i < j < k \leq n$ (2.8e)

$0 \leq x_{ij} \leq 1, \ \forall i, j: \quad 1 \leq i < j \leq n.$ (2.8f)

For the sake of simplicity, we used unweighted graphs for our test problems, that is $c \equiv e$, as in Barahona and Anbil (2000), which yet yields a class of challenging problem instances. The number of vertices n was selected as 40, 50, 60, 70, and 80. The specifications for the resulting test bed of problems are displayed in Table 2.1, along with the effort required by CPLEX 8.1 (on a Pentium III 667 Mhz computer) to solve these linear relaxations.

Transportation Problems

Assuming n sources and n sinks, the standard transportation problem can be formulated as follows.

TP: Minimize $c^T x$ (2.9a)

subject to $\sum_{j=1}^{n} x_{ij} = s_i, \quad \forall\ i = 1,...,n$ (2.9b)

$$\sum_{i=1}^{n} x_{ij} = d_j, \qquad \forall\ j = 1,...,n, \tag{2.9c}$$

$$x_{ij} \geq 0, \qquad \forall\ i, j = 1,...,n,$$

where s_i, $i = 1,...,n$, and d_j, $j = 1,...,n$, represent the supplies and the demands, respectively, and c is the transportation cost vector. By relaxing constraints (2.9b) and simplifying the subproblem, we obtain the dual problem given by

LDTP: $\underset{\pi \in R^m}{\text{Maximize}} \quad \theta(\pi)$, (2.10a)

$$\text{where } \theta(\pi) = \sum_{i=1}^{n} s_i \pi_i + \sum_{j=1}^{n} d_j \min_i \{c_{ij} - \pi_i\}. \tag{2.10b}$$

Note that TP in (2.9) has equality constraints and only the supply balancing constraints in (2.9b) are dualized in LDTP, and hence, we now have $\Pi = R^n$. Furthermore, note that θ in (2.10b) is derived from the Lagrangian subproblem that has only equality constraints in (2.9c).

For the primal solution, a random basis B having arcs (i, j) belonging to a tree graph was first generated by starting with some random arc $(p,q) \in B$, and defining $N = \{\text{source } p, \text{and sink } q\}$ and $\overline{N} = \{\text{remaining sources and sinks}\}$. Then at each subsequent step, some random arc (from the bipartite transportation graph) in the cut-set $(N, \overline{N})$ was selected to be included in B with the corresponding source/sink from $\overline{N}$ being transferred to N, until $\overline{N}$ became empty after $2n-2$ steps. The flows on $(i, j) \in B$ were then randomly generated as integers from a uniform discrete distribution on [0,10]. Some ρ_p % of these basic flows were made zeros to induce a desired extent of primal degeneracy. The dual solution $(u,-v)$, having variables u and $-v$ associated with constraints (2.9b) and (2.9c), respectively, was generated with its components distributed uniformly on [-5,5]. The right-hand-side vectors s and d were calculated by (2.9b) and (2.9c), respectively, based on the prescribed primal solution. The cost vector c was derived based on the prescribed dual solution by letting $c_{ij} = u_i - v_j$, $\forall (i, j) \in B$, and

Table 2.2 Summary of Transportation Test Problems.

Problem	Number of sources / destinations	Number of variables	Number of constraints	CPLEX solution	
				Objective value	CPU time (sec)
TR1	400	160000	800	2318.66	131.47
TR2	500	250000	1000	-2905.57	222.96
TR3	600	360000	1200	909.87	447.47
TR4	700	490000	1400	-423.76	757.65
TR5	800	640000	1600	-1717.66	1126.94

$c_{ij} = u_i - v_j + r_{ij} \;\; \forall (i,j) \notin B$, where r_{ij} was selected as some random variable on [0,5]. Some $\rho_D\%$ of these dual slacks were enforced to be zero to induce a desired extent of dual degeneracy. Note that ρ_P and ρ_D respectively control the extent of primal and dual degeneracy. Five test problems having the number of sources and sinks each equal to 400, 500, 600, 700, and 800 were generated for (ρ_P , ρ_D) = (10%, 10%) as summarized in Table 2.2.

General Linear Programming Problems

In addition to the above test problems, we used linear programming test problems having the form given by

LP: Minimize $c^T x$ (2.11a)
subject to $Ax \geq b$ (2.11b)
$x \in X \equiv \{x : 0 \leq x \leq e\}$. (2.11c)

Note that LP is different from LR in that it assumes that the structural constraints of the type $Dx \geq d$ have been accommodated within $Ax \geq b$. This simplifies the solution of the subproblems, while sacrificing a degree of control within the Lagrangian dual optimization process.

For the primal solution, some $\min\{n, \lfloor 0.75m \rfloor\}$ components of x were randomly generated on $[0,1]$, with some $\rho_P\%$ of these being set at 0 or 1, while the rest of the x variables were necessarily randomly set to 0 or 1. Furthermore, $m - \min\{n, \lfloor 0.75m \rfloor\}$

Table 2.3 Summary of LP Test Problems.

(ρ_P, ρ_D)	Prob	Rows/ Columns	CPLEX solution		Prob	Rows/ Columns	CPLEX solution	
			Objective value	CPU time (sec)			Objective value	CPU time (sec)
(5%, 5%)	LP1	500/1000	-2146.43	28.03	LP6	1000/500	-280.01	26.36
	LP2	1000/3000	-2747.79	571.56	LP7	3000/1000	-474.71	559.05
	LP3	1000/5000	-5579.55	829.07	LP8	5000/1000	-1626.49	863.67
	LP4	2000/3000	-2419.05	6651.61	LP9	3000/2000	-3965.08	5950.16
	LP5	2000/5000	-6584.23	7958.14	LP10	5000/2000	-2656.42	9619.07
(5%, 25%)	LP11	500/1000	-156.64	23.23	LP16	1000/500	-127.55	28.02
	LP12	1000/3000	-2130.92	468.9	LP17	3000/1000	1517.06	636.37
	LP13	1000/5000	-324.37	739.97	LP18	5000/1000	1205.84	902.43
	LP14	2000/3000	-864.19	5018.73	LP19	3000/2000	-1326.20	6524.15
	LP15	2000/5000	-2359.49	19648.1	LP20	5000/2000	2596.23	13844.1
(25%, 5%)	LP21	500/1000	-1225.18	24.36	LP26	1000/500	139.09	23.66
	LP22	1000/3000	-2866.79	573.94	LP27	3000/1000	-813.59	527.3
	LP23	1000/5000	-5124.29	794.23	LP28	5000/1000	-577.26	710.34
	LP24	2000/3000	-1596.25	5582.36	LP29	3000/2000	342.24	4640.3
	LP25	2000/5000	-4023.94	7619.21	LP30	5000/2000	3738.78	6924.76
(25%, 25%)	LP31	500/1000	-1581.77	22.41	LP36	1000/500	-211.00	23.91
	LP32	1000/3000	-2356.31	520.07	LP37	3000/1000	932.31	561.17
	LP33	1000/5000	-3141.28	765.4	LP38	5000/1000	-1646.98	676.3
	LP34	2000/3000	-961.15	4837.27	LP39	3000/2000	3771.27	4290.73
	LP35	2000/5000	6058.19	7184.66	LP40	5000/2000	4082.77	6988.49

slack variables in (2.11b), denoted by s, were generated from a uniform distribution on [0,5], again with some ρ_P% of them being set at zero, while the rest of these slack variables were necessarily set to 0. Let $(\bar{x}, \bar{s})$ denote the resulting solution. The coefficient matrix A was generated based on a uniform distribution on [-5,5], having a density of 0.05, and modified in order to have at least one nonzero component in each row. The right-hand-side vector b was chosen as $b = A\bar{x} - \bar{s}$.

For the dual solution π associated with constraints (2.11b), the components π_i, which correspond to the slack variables s_i that were necessarily set equal to zero, were

uniformly generated on [0,10], with some $\rho_D\%$ of these being set equal to zero. The remaining components of π were set to zeros. The dual variables γ and μ corresponding to the lower and upper bounding constraints in (2.11c), respectively, were generated as follows. Corresponding to the x_j variables that were necessarily set at zero above, we put $\mu_j = 0$ and generated γ_j uniformly on [0,5], randomly setting some $\rho_D\%$ of these at zeros. Likewise, corresponding to the x_j variables that were necessarily set at one above, we put $\gamma_j = 0$ and generated μ_j uniformly on [0,5], randomly setting some $\rho_D\%$ of these at zero. The remaining γ and μ variable components were set at zero. Letting $(\bar{\pi}, \bar{\gamma}, \bar{\mu})$ be the resulting dual solution, we computed the coefficient vector of the objective function, c, as $c = \bar{\pi}A + \bar{\gamma} - \bar{\mu}$. We generated sets of forty test problems having various sizes for each (ρ_P, ρ_D) = (5%, 5%), (5%, 25%), (25%, 5%), and (25%, 25%). A summary of the test problems thus generated is displayed in Table 2.3, along with their solution performance using CPLEX 8.1.

Set Covering Problems

Given a set S and its subsets, the set covering problem seeks a minimal sized collection of subsets that covers S. The linear programming relaxation of a set covering problem (SC) can be formulated as follows.

SC: Minimize $c^T x$

subject to $Ax \geq e$

$0 \leq x \leq e$,

where A has 0-1 coefficients, and where e is an appropriately dimensioned vector of ones. A summary of five test problems taken from Beasley (1990) are displayed in Table 2.4.

Computational Results

The algorithmic strategies VA, RVA, BVA, ADS, and PKC used in concert with VTVM as described in Section 2.3 were coded and compiled on a Visual C++ Compiler

Table 2.4 Summary of Set Covering Problems.

Problem	Number of variables	Number of constraints	Density of coefficient matrix	CPLEX solution	
				Objective Value	CPU time (sec)
RAIL507	63009	507	0.013	172.15	116.65
RAIL2586	920683	2586	0.0034	935.92	24081
RAIL4284	1092610	4284	0.0024	1054.05	44243.7

6.0. All runs were made on a Pentium III 667Mhz computer. The matrix data was input in a CSC (Compressed Sparse Column) format, which is designed for storing general sparse matrices. The maximum number of iterations was set to $k_{\max} = 2000$. The algorithmic parameters were initialized as in Remarks 2.2 through 2.5, and the initial dual solution was set as a zero vector. For the purpose of comparing the quality of solutions produced and the empirical ability of each algorithm to close the duality gap, we use a statistic, *percentage optimality ratio* (POR), that is defined as the ratio: [optimal objective value minus the best detected objective value] ÷ [optimal objective value minus the initial objective value] $\times 100\%$, to record the gap-closing performance of each algorithm. (For example, if $\theta_1 = -18018.6$, $\theta^* = 2318.66$, and $\theta_{best} = 2045.79$, the corresponding percentage optimality ratio is computed as $\Gamma = 100 \times (\theta^* - \theta_{best}) / (\theta^* - \theta_1) = 1.34\%$.) Note that only the value θ_{best} here varies from one algorithm to another. The resulting percentage optimality ratios for each test problem are exhibited in Table 2.5. In addition, aggregated results are displayed in Table 2.6, which includes average percentage optimality ratios for each problem type, an overall average percentage optimality ratio, and a maximum percentage optimality ratio.

For the max cut problems, all algorithms produced the optimal objective values within 1000 iterations. For the transportation problems, the algorithms yielded a distinctive ranking in the order of PKC, VA, ADS, RVA, and BVA with respect to worsening percentage optimality ratios. Moreover, the average percentage optimality ratio for RVA was 36.55% less than that for BVA (Table 2.6), while RVA itself did not yield solutions as good as those obtained by the other three methods, resulting in a percentage optimality ratio that was 19.22% greater than that for PKC. It is worth

Table 2.5 Percentage Optimality Ratio Values for Each Test Problem (%).

Problem	VA	RVA	BVA	ADS	PKC	Problem	VA	RVA	BVA	ADS	PKC
MC1	0	0	0.02	0	0	TR1	1.34	20.4	74.38	4.12	0.81
MC2	0	0	0	0	0	TR2	0.96	20.27	46.62	1.91	0.55
MC3	0	0	0	0	0	TR3	3.62	20.27	30.14	2.69	1.31
MC4	0	0	0	0	0	TR4	1.77	7.24	64.01	3.16	1.53
MC5	0	0	0	0	0	TR5	2.69	33.32	69.11	5.04	1.19
LP1	0.01	0	0	0	0.01	LP21	0.01	0	0	0	0.01
LP2	0	0	0	0	0	LP22	0	0	0	0	0
LP3	0	0	0	0	0	LP23	0	0	0	0	0
LP4	0.01	0	0	0	0.01	LP24	0.02	0	0	0	0.01
LP5	0	0	0	0	0	LP25	0	0	0	0	0
LP6	0.26	0.15	0.22	0.13	0.24	LP26	0.09	0.36	2.17	0.1	0.13
LP7	0.32	0.1	0.21	0.17	0.27	LP27	0.2	1.63	6.15	0.52	0.15
LP8	0.37	0.62	2.32	0.44	0.35	LP28	0.23	1.46	2.62	0.23	0.18
LP9	0.3	0.05	0.06	0.12	0.23	LP29	0.14	0.02	0.02	0.07	0.14
LP10	0.36	0.05	0.07	0.19	0.3	LP30	0.29	0.42	2.02	0.53	0.23
LP11	0	0	0	0	0	LP31	0.01	0	0	0	0.01
LP12	0	0	0	0	0	LP32	0	0	0	0	0
LP13	0	0	0	0	0	LP33	0	0	0	0	0
LP14	0.01	0	0	0	0.01	LP34	0.01	0	0	0	0.01
LP15	0	0	0	0	0	LP35	0	0	0	0	0
LP16	0.3	0.24	0.42	0.17	0.3	LP36	0.2	0.31	0.42	0.18	0.22
LP17	0.38	0.09	0.15	0.24	0.3	LP37	0.2	0.43	3.55	0.25	0.17
LP18	0.41	0.58	1.65	0.25	0.44	LP38	0.18	1.49	6.43	0.39	0.2
LP19	0.35	0.06	0.05	0.2	0.39	LP39	0.21	0.04	0.05	0.13	0.21
LP20	0.39	0.07	0.06	0.22	0.35	LP40	0.33	1.01	2.2	0.76	0.3
RAIL506	0.41	2.02	6.07	0.2	0.39						
RAIL2596	0.34	3.1	1.16	0.75	0.34						
RAIL4284	0.41	8.09	6.26	0.41	0.25						

pointing out here that the Lagrangian dual of transportation problems is known to produce challenging instances for subgradient-based approaches (see Shor, 1985).

For the general linear programming problems, we observed that the ADS method revealed a slightly better performance than the VA strategy for the problems having m and n such that $m < n$. Note that, for these problems, ADS outperformed VA for 17 instances out of 20, while there were no significant differences between the two methods for the remaining three problems, LP12, LP13, and LP33. The problems having $m > n$ proved to be somewhat more challenging for these dual-based methods as might be

Table 2.6 Average and Maximum POR Values.

Average for	VA	RVA	BVA	ADS	PKC
MC	0	0	0	0	0
TR	2.08	20.3	56.85	3.38	1.08
LP	0.14	0.23	0.77	0.13	0.13
SC	0.39	4.4	4.5	0.45	0.33
All Problems	0.38	2.34	6.2	0.44	0.22
Maximum POR for All Problems	3.62	33.32	74.38	5.04	1.53

expected, for which the ADS strategy yielded relatively better results than VA for 13 instances out of 20. Over the LP problems, ADS and PKC yielded a POR value of 0.13% on average, followed closely by VA, which revealed the next best POR value of 0.14% (Table 2.6). Finally, for set covering problems, we observed that the performance of VA was better than that of ADS, except for the relatively smaller-sized problem RAIL507, while the PKC strategy yielded the smallest average percentage optimality ratio that was slightly lesser (by 0.06%) than that for VA, with ADS coming in third by the same amount. On the other hand, RVA and BVA produced percentage optimality ratios of values about 4% greater than that for ADS.

From the summary of results given in Table 2.6, it is evident that PKC exhibited the best performance for all types of test problems, and that the VA strategy was second best, yielding a 0.16% greater percentage optimality ratio at an average over all the problems. The ADS strategy came a close third, yielding at an average a 0.06% greater percentage optimality ratio than VA. However, the RVA and BVA strategies were significantly worse (average percentage optimality ratio exceeding that for ADS by 1.9%), with RVA revealing a better performance (3.86% smaller percentage optimality ratio on average) than BVA. This result agrees with the experiment of Bahiense et al. (2002). Moreover, in the light of robustness, observe that the maximum percentage optimality ratios given in Table 2.6 are compatible in rank-order with the average percentage optimality ratio values.

The CPU effort for each algorithm, as well as that for the CPLEX 8.1 solver, are displayed in Table 2.7. For a meaningful comparison, let us define a *near-optimal*

Table 2.7 CPU Times (seconds) to Attain Near-Optimality and CPLEX Run Times.

Prob	VA	RVA	BVA	ADS	PKC	CPLEX	Prob	VA	RVA	BVA	ADS	PKC	CPLEX
MC1	17.78	4.14	28.73	24.05	35.27	101.01	TR1	8.95	*MAX*	*MAX*	9.56	9.52	131.47
MC2	36.39	64.2	53.39	39.45	68.05	368.65	TR2	14.41	*MAX*	*MAX*	18.3	15.89	222.96
MC3	66.36	84.11	70.5	33.62	118.94	1137.94	TR3	28.3	*MAX*	*MAX*	33.02	23.06	447.47
MC4	117.01	143.7	*MAX*	140.56	201.82	3561.68	TR4	39.31	*MAX*	*MAX*	45.5	31.94	757.65
MC5	163.2	288.26	130.78	204.92	312.6	62253.5	TR5	52.86	*MAX*	*MAX*	*MAX*	43.8	1126.94
LP1	1.36	1.7	1.36	1.8	1.36	28.03	LP21	1.52	2.02	1.52	2.02	1.67	24.36
LP2	8.34	10.19	9.39	10.53	8.97	571.56	LP22	8.91	10.58	8.37	11	9	573.94
LP3	12.59	17.75	15.95	19.33	13.66	829.07	LP23	12.77	17.16	18	19.17	13.97	794.23
LP4	17.08	23.23	21.63	23.89	18.3	6651.61	LP24	16.91	22.48	20.41	23.88	17.44	5582.36
LP5	27.92	37.84	33.41	39.36	28.33	7958.14	LP25	27.58	38.01	32.77	41.14	27.3	7619.21
LP6	1.52	1.41	1.23	2	1.59	26.36	LP26	1.49	1.8	*MAX*	2.03	1.72	23.66
LP7	9.75	10.31	8.3	12.91	10.28	559.05	LP27	10.34	13.27	*MAX*	14.48	10.84	527.3
LP8	19.61	19.75	*MAX*	25.02	21.09	863.67	LP28	18.22	*MAX*	*MAX*	23.52	18.66	710.34
LP9	17.3	24.73	22.98	22.94	18.05	5950.16	LP29	17.28	28.08	24.33	23.58	18.22	4640.3
LP10	32.87	37.5	33.67	40.61	33.19	9619.07	LP30	34.16	42.53	*MAX*	45.51	34.92	6924.76
LP11	1.58	1.36	1.33	2.2	1.75	23.23	LP31	1.61	1.83	1.7	2.17	1.61	22.41
LP12	8.25	9.52	9.2	10.55	8.73	468.9	LP32	7.77	8.86	9.23	10.72	8.23	520.07
LP13	14.63	15.28	15.02	19.27	15.3	739.97	LP33	14.33	16.19	15.97	19.2	14.97	765.4
LP14	17.98	21.53	20.7	23.55	18.27	5018.73	LP34	17.03	20.31	19.72	24.12	17.39	4837.27
LP15	31.92	29.91	32.97	39.81	31.61	19648.1	LP35	26.78	33.72	32.95	39.06	32.45	7184.66
LP16	1.69	1.66	1.52	2.03	1.86	28.02	LP36	1.63	1.67	1.67	2.13	1.81	23.91
LP17	10.8	9.47	8.78	13.45	11.28	636.37	LP37	10.48	*MAX*	*MAX*	12.78	10.91	561.17
LP18	20.12	18.31	*MAX*	23.27	21.05	902.43	LP38	19.27	*MAX*	*MAX*	23.42	20.61	676.3
LP19	17.5	25.98	25.66	25.08	18.89	6524.15	LP39	17.5	24.48	23.53	24.03	18.31	4290.73
LP20	34.08	38.69	36.56	43.28	35.03	13844.1	LP40	33.48	55.42	*MAX*	51.09	35.08	6988.49
RAIL506	22.83	*MAX*	*MAX*	19.34	25.66	116.65	*MAX*: failed to attain the corresponding near-optimal value.						
RAIL2596	408.58	*MAX*	*MAX*	365.62	424.82	24081							
RAIL4284	678.95	*MAX*	*MAX*	529.3	730.28	44243.7							

Table 2.8 Average Relative CPU Percentage.

Average for	VA	RVA	BVA	ADS	PKC
MC	7.37	6.68	29.87	8.35	14.00
TR	5.89	100	100	25.77	5.52
LP	2.15	9.69	24.12	2.83	2.29
SC	7.60	100	100	6.43	8.47
All Problems	3.30	23.04	36.12	5.72	4.05

objective value as the objective value corresponding to a percentage optimality ratio of 1% for the max cut, general linear programming, and set covering problems, and a percentage optimality ratio of 5% for the relatively more difficult class of transportation problems. A run that could not yield a near-optimal objective value within 2000 iterations is marked as *MAX*, and the least effort to attain a near-optimal objective value is shaded.

While the VA strategy required the least effort among the different algorithms for a majority of test problems (36 of 53 instances), we observe that PKC consumed CPU times similar to VA except for the max cut problems. Although BVA yielded the second best performance in terms of the number of instances (7 instances) for which the algorithm spent the smallest CPU time, it could not attain a near-optimal solution for eighteen test problems. Furthermore, note that RVA and BVA were unable to find solutions having a near-optimal objective value within 2000 iterations for the transportation and set covering problems. For the transportation problems, VA required less CPU times for smaller sized problems, while PKC consumed the least effort as the problem size increased (TR3, TR4, and TR5). For the set covering problems, the ADS strategy consumed 10-22% less CPU times than VA, the second best algorithm.

For an insightful comparison of overall computational effort, we computed a *relative CPU percentage* defined by [the CPU time consumed until a near-optimal objective value is attained] ÷ [the CPLEX run time to solve the corresponding problem]×100% for each test problem. The averages of these values for each problem type and algorithmic strategy are summarized in Table 2.8. To assign some penalty for not being able to attain a near-optimal objective value for any particular instance, we ascribed a relative CPU percentage of 100%. Note that VA consistently revealed a good performance (best for the general linear programming problems and second best for the

other classes of test problems). PKC was the best for transportation problems and was a close second best for general linear programming problems. Overall, we conclude that the convergence rate of the VA strategy is the best among the tested algorithms.

Finally, we observed that CPLEX consumed greater effort than VA by a factor of 5.1 to 615.5, although CPLEX determined optimal solutions while the average percentage optimality ratio gap for VA was 0.38%. Furthermore, as expected, the CPU time consumed by CPLEX was exorbitantly lengthened as the problem size increased. For example, as the number of vertices for the max cut problem was doubled from 40 (MC1) to 80 (MC5), the CPU time for CPLEX increased by 61529% while the effort to attain a near-optimal solution increased by 818%, 752%, and 786%, for VA, ADS, and PKC, respectively.

2.6 Summary and Conclusions

We have adapted the Volume Algorithm (VA) of Barahona and Anbil (2000) and have embedded it within a variable target value method VTVM (Sherali et al, 2000) as a direction-finding strategy. This adaptation makes VA resemble a deflected subgradient scheme in contrast with the bundle type interpretation of the modification of VA as afforded by Bahiense et al. (2002). Furthermore, we have established convergence of the algorithm under the condition that the geometric moving average parameter α and the step-length factor β satisfy $0 < \beta \leq \alpha \leq 1$. Moreover, this affords generic implementation of the VA strategy without the need for devising specialized heuristics for computing upper bounds, tailored for each specific class of problems.

Using a variety of test problems, we evaluated the performance of the VA strategy within this VTVM framework and compared this against an alternative deflection scheme ADS (Sherali and Ulular, 1989), a cutting-plane strategy PKC (Sherali et al., 2001), and the modified RVA and BVA procedures (Bahiense et al., 2002), all embedded within the same VTVM framework. In terms of the quality and robustness of solutions produced by each algorithm within 2000 iterations, the PKC strategy revealed the best performance among all implemented algorithms for our test bed of problems, with the VA and the ADS strategies coming in a relatively close second and third, respectively. As far as CPU times are concerned, the VA strategy consumed the least computational effort for most

problems to attain a near-optimal objective value, while PKC was the second best. Moreover, the VA, ADS, and PKC strategies revealed a considerable savings in CPU effort over CPLEX 8.1 when used to derive near-optimal solutions. This holds good promise, therefore, in implementing such strategies for deriving quick lower bounds via LP relaxations for solving relatively large-scale MIP problems.

Although the Volume Algorithm was originally designed for recovering a primal solution, its primal convergence is not guaranteed as exhibited by our counter-example in Section 2. It would be worthwhile to modify the algorithm to achieve primal convergence. Guidelines for approaching such an analysis are inherent in the dual convergence analysis presented herein, and the convergent primal recovery processes described in Shor (1985), Sherali and Choi (1996), and Larsson et al. (1999). We propose this for future research, along with a study of adopting the procedures explored herein (particularly, the VA, PKC, and ADS strategies) in lieu of traditional LP solvers within the context of branch-and-bound/cut methods for solving mixed-integer programs.

Chapter 3:
Convergence and Computational Analyses for Two Variable Target Value Methods: VTVM and the Level Algorithm

3.1 Introduction

We consider two target value frameworks that do not require any prior knowledge of an upper bound on the optimal objective function value: The Level Algorithm and VTVM. Brännlund (1993) designed the Level Algorithm, in which the target value (or *level*) update is made either when the objective value exceeds a specified threshold, or the distance traversed without experiencing a sufficient improvement reaches a prescribed tolerance. Goffin and Kiwiel (1999) proved that the Level Algorithm is convergent to an optimum when used along with the pure subgradient strategy. Sherali et al. (2000) proposed the variable target value method (VTVM), where the target value is revised based on the recent history of improvements. They established the convergence of VTVM along with a general deflection scheme that subsumes the pure subgradient strategy.

This chapter focuses on the convergence analysis and numerical performance of two variable target value methods when implemented in conjunction with several direction finding and step-length strategies. Furthermore, some modifications for these variable target frameworks are suggested to improve their computational effectiveness. In Section 3.2, we describe the details of these two variable target value algorithms, along with alternative competitive direction finding and step-length subroutines such as the pure subgradient (PS) method of Polyak (1969), the average direction strategy (ADS) of Sherali and Ulular (1989), the Volume Algorithm (VA) strategy discussed in Chapter 2, and a generalization of the modified Polyak-Kelly cutting-plane method (PKC) of Sherali et al. (2001) (which we now call GPKC). While the convergence of the simple Level Algorithm is addressed in Goffin and Kiwiel (1999), we further provide in Section 3.3 a

convergence analysis of the Level Algorithm when used in concert with the aforementioned direction finding and step-length strategies: ADS, VA, and GPKC. In addition, the convergence of the GPKC strategy in the VTVM framework is established. We also discuss some potential weaknesses of the above two target value updating schemes, and suggest modifications to improve them while preserving their convergence properties in Section 3.4. A computational study of these algorithmic procedures is provided in Section 3.5 using various test problems. Finally, concluding remarks and extensions for further study are provided in Section 3.6.

3.2 Evaluated Algorithmic Procedures

In this section, we outline the various algorithms that we investigate for theoretical convergence and computational performance. The Level Algorithm of Brännlund (1993) and the VTVM approach of Sherali et al. (2000) were used as the fundamental frameworks for implementing the target value update process. The following notation is used for describing all these algorithmic procedures.

k ≡ iteration counter;

π^k ≡ k th dual iterate ;

θ_k ≡ $\theta(\pi^k)$;

x^k ≡ solution to the Lagrangian subproblem at π^k ;

g^k ≡ $b - Ax^k$ (a subgradient at π^k);

d^k ≡ search direction at iteration k ;

$\bar{\pi}$ ≡ best dual solution upto iteration k (incumbent solution);

z ≡ $\theta(\bar{\pi})$ (incumbent objective value);

$\bar{g}$ ≡ subgradient at $\bar{\pi}$;

l ≡ target value adjustment (or outer iteration) counter ;

w_l ≡ target value at outer iteration l ;

$P_\Pi(\cdot)$ ≡ orthogonal projection onto Π .

Variable Target Value Frameworks

The Level Algorithm of Brännlund (1993), and the VTVM procedure of Sherali et al. (2000) adopt a similar basic scheme in updating the target value. In both algorithms, the target value is increased when a significant improvement is achieved, and is decreased when a specified effort has been expended without obtaining some sufficient extent of improvement. Furthermore, whenever the target value is decreased, the methods are restarted from the incumbent solution ($\overline{\pi}$). In the VTVM algorithm, the target value is revised based on the recent history of improvements whenever it is increased following a sufficient improvement, or it is revised to equal an average of the previous target value and a minimal targeted improvement level whenever it is decreased following a specified number of iterations during which it fails to attain a designated improvement. The improvement acceptance tolerance parameter is accordingly reset as some fraction of the difference between the revised target value and the incumbent objective value, or some prescribed threshold value, whichever is greater. On the other hand, in the Level Algorithm, the improvement acceptance tolerance parameter is initialized at some arbitrary value, and is halved each time the algorithm traverses a threshold distance in the dual space without being able to at least halve the gap between the target value and the incumbent objective value that was recorded at the iteration when the previous adjustment of the target value was performed. Whenever this adjustment is made, or the required improvement is attained, the new target value is accordingly revised to a value equal to the incumbent solution value plus the revised acceptance tolerance. The Level Algorithm, modified for computational implementation as indicated below, and the VTVM procedure operate as follows.

<u>Level Algorithm</u>

Initialization Step: Select a termination tolerance $\varepsilon_0 > 0$ on subgradient norms, a termination criterion $k_{\max}$ on the maximum number of iterations, a target value parameter tolerance $\sigma_1 > 0$, and a tolerance $R > 0$ on the distance traversed without a sufficient improvement. Set the cumulative distance measure $\Delta_1 = 0$, and the reset indicator RESET=1. (Select integers $\overline{\delta}' \geq 0$ and $\overline{\delta}'' \geq 1$ for Subroutine GPKC specified below.)

Select an initial solution $\pi^1 \in \Pi$. Compute θ_1, g^1, and put $z_1 = \theta_1$, $\bar{\pi} = \pi^1$, and $\bar{g} = g^1$. If $\|g^1\| < \varepsilon_0$, terminate the algorithm. Initialize the iteration counter $k = 1$, the adjustment counter $l = 1$, and $k(l) = 1$, where $k(l)$ is the iteration number corresponding to the l^{th} adjustment of the target value. Set the target value $w_l = z_{k(l)} + \sigma_l$, and select an initial value for the step-length parameter $\beta > 0$.

STEP 1: Call a subroutine that implements a direction and step-length strategy to obtain an intermediate iterate $\hat{\pi}^{k+1}$ and the new iterate $\pi^{k+1} \equiv P_\Pi(\hat{\pi}^{k+1})$. Set $\Delta_{k+1} = \Delta_k + \|\hat{\pi}^{k+1} - \pi^k\|$ and increment $k \leftarrow k+1$. Determine θ_k and $g^k \in \partial\theta(\pi^k)$. If $\theta_k > z_{k-1}$, then set $z_k = \theta_k$, $\bar{\pi} = \pi^k$, $\bar{g} = g^k$. Otherwise, set $z_k = z_{k-1}$. Put RESET=0. If $k > k_{\max}$ or $\|g^k\| < \varepsilon_0$, terminate the algorithm.

STEP 2: If $\theta_k < z_{k(l)} + \frac{1}{2}\sigma_l$, go to Step 4. Otherwise, set $\sigma_{l+1} = \sigma_l$, $k(l+1) = k$, $\Delta_k = 0$, and proceed to Step 3.

STEP 3: Compute $w_{l+1} = z_{k(l+1)} + \sigma_{l+1}$, increment $l \leftarrow l+1$, and return to Step 1.

STEP 4: If $\Delta_k \le R$, return to Step 1. Otherwise, set $\sigma_{l+1} = \frac{1}{2}\sigma_l$. Put $\pi^k = \bar{\pi}$, $g^k = \bar{g}$, and RESET=1. Set $k(l+1) = k$ and $\Delta_k = 0$. Compute $w_{l+1} = z_{k(l+1)} + \sigma_{l+1}$. If $(w_l - w_{l+1}) \le 0.1$, then set $\beta \leftarrow \max\{\beta/2,\ 10^{-6}\}$. Increment $l \leftarrow l+1$ and return to Step 1.

VTVM Algorithm

Initialization Step: Select a termination tolerance $\varepsilon_0 > 0$ on subgradient norms, a threshold value $\varepsilon > 0$ for the improvement acceptance tolerance parameter, the maximum number of iterations $k_{\max}$, an initial step-length parameter $\beta \in (0, 1]$, a factor $\sigma \in (0, 1/3]$ for updating the target value, the maximum number of iterations $\bar{\tau}$ without attaining a significant improvement, the maximum number of consecutive non-improving iterations $\bar{\gamma}$ without improvement, and set an algorithmic parameter $\eta \in (0, 1]$ related to increasing the target value. (Select integers $\bar{\delta}' \ge 0$ and $\bar{\delta}'' \ge 1$ for Subroutine GPKC

specified below.)

Select an initial solution $\pi^1 \in \Pi$, and determine θ_1 and g^1. Set $\bar{\pi} = \pi^1$, $\bar{g} = g^1$, and $z_1 = \theta_1$. If $\|g^1\| < \varepsilon_0$, terminate the algorithm. Initialize iteration counters $l = 1$ and $k = 1$. Put the iteration counter without attaining the targeted level of improvement $\tau = 0$, the iteration counter for consecutive non-improvements $\gamma = 0$, the partial cumulative amount of improvement $\Delta = 0$, and the reset indicator RESET=1. Initialize the target value $w_1 = \min\left\{UB, \theta_1 + \|g^1\|^2 / 2\right\}$ where *UB* is some known upper bound on the problem (possibly infinite). Let the acceptance tolerance $\varepsilon_1 = \sigma(w_1 - \theta_1)$.

STEP 1: Call a subroutine that implements a direction and step-length strategy to obtain π^{k+1}. Increment $\tau \leftarrow \tau + 1$ and $k \leftarrow k + 1$. Compute θ_k and $g^k \in \partial\theta(\pi^k)$. Put RESET=0.

STEP 2: If $\theta_k > z_{k-1}$, set $\Delta \leftarrow \Delta + (\theta_k - z_{k-1})$, $z_k = \theta_k$, $\bar{\pi} = \pi^k$, $\bar{g} = g^k$, $\gamma = 0$, and proceed to Step 3. Otherwise, put $z_k = z_{k-1}$, increment $\gamma \leftarrow \gamma + 1$, and go to Step 4.

STEP 3: If $k > k_{\max}$ or $\|g^k\| < \varepsilon_0$, terminate the algorithm. If $z_k \geq w_l - \varepsilon_l$, go to Step 5. If $\tau \geq \bar{\tau}$, go to Step 6. Else, return to Step 1.

STEP 4: If $k > k_{\max}$, terminate the algorithm. If $\gamma \geq \bar{\gamma}$ or $\tau \geq \bar{\tau}$, go to Step 6. Else, return to Step 1.

STEP 5: Compute $w_{l+1} = z_k + \varepsilon_l + \eta\Delta$. Let $\varepsilon_{l+1} = \max\{(w_{l+1} - z_k)\sigma, \varepsilon\}$, put $\tau = 0$, $\Delta = 0$, and increment $l \leftarrow l + 1$. Return to Step 1.

STEP 6: Compute $w_{l+1} = \dfrac{(z_k + \varepsilon_l) + w_l}{2}$, and $\varepsilon_{l+1} = \max\{(w_{l+1} - z_k)\sigma, \varepsilon\}$. If $\gamma \geq \bar{\gamma}$, then set $\bar{\gamma} \leftarrow \min\{\bar{\gamma} + 10, 50\}$. If $(w_l - w_{l+1}) \leq 0.1$, then set $\beta \leftarrow \max\{\beta / 2,\ 10^{-6}\}$. Put $\gamma = 0$, $\tau = 0$, $\Delta = 0$, and $l \leftarrow l + 1$. Reset $\pi^k = \bar{\pi}$, $\theta_k = z_k$, $g^k = \bar{g}$, and put RESET=1. Return to Step 1.

Remark 3.1: In the Level Algorithm, some modifications have been made for computational implementation. Besides the stopping criterion $g^k = 0$ (or practically,

$\left\| g^k \right\| < \varepsilon_0$), we terminate the algorithm when the iteration counter reaches some specified limit, $k_{\max}$. Furthermore, the step-length parameter β is updated at Step 4 whenever the target value decrement is less than 0.1 as in Sherali et al. (2000). For an improved implementation of the VTVM algorithm that preserves convergence properties, as prescribed by Sherali et al. (2000), we initialize $r \in (0,1)$ and $\bar{r} = r + 1$, and we increase the target value at Step 5 according to $w_{l+1} = z_k + \max\{\varepsilon_l + \eta\Delta, r|z_k|\}$, where r is updated by $r \leftarrow r / \bar{r}$ whenever the second term in the maximand is greater.

Remark 3.2: The parameter values used are as follows. In both algorithms, we set $\varepsilon_0 = 10^{-6}$ and $k_{\max}$ = 2000 for terminating the algorithm, and we select $\beta = 0.8$ for the initial step-length parameter. In the Level Algorithm, we set $R = 10^4$ and initialize the target value parameter tolerance $\sigma_1 = \left\| g^1 \right\| R / 2 = \left\| g^1 \right\| \times 5000$ as implemented in Goffin and Kiwiel (1999) for solving TSP problems. For the VTVM algorithm, we use $\varepsilon = 0.1$, $\sigma = 0.15$, $\eta = 0.75$, $r = 0.1$, $\bar{\tau} = 75$, and $\bar{\gamma} = 20$ as recommended in Sherali et al. (2000). Furthermore, for both algorithms, the initial solution is chosen as $\pi^1 = 0$.

Direction Finding and Step-Length Strategies

In the pure subgradient (PS) algorithm, the direction of search is simply taken as a subgradient vector that is generated at the current solution. The step-length to adopt along this subgradient direction in the present context is determined by substituting the current target value for the optimal objective value in the step-length prescribed by Polyak (1969). In order to improve the convergence behavior, the average direction strategy (ADS) of Sherali and Ulular (1989) deflects the subgradient direction by bisecting the angle between it and the previous direction. The Volume Algorithm of Barahona and Anbil (2000) uses a geometric moving average of all previous subproblem solutions to compute search directions from *stability centers*. These geometric weights are considered to be estimates for the solution to the corresponding Danzig-Wolfe decomposition of LP. In Chapter 2, we embedded the Volume Algorithm (VA) as a variant of a deflected subgradient scheme within the VTVM framework, and exhibited that this strategy yields

competitive computational results.

In addition, we generalize the modified Polyak-Kelly cutting plane (PKC) method of Sherali et al. (2001), which performed best in comparison with several competing space dilation type strategies. To be more specific, let us define $\delta \geq 0$ by $\delta \equiv k - \bar{k}$, where $\bar{k}$ is the most recent iteration at which the variable target value algorithm was reset, i.e., RESET=1. Denoting θ^* as the optimal dual value, we have by the concavity of that $\theta(\pi) \leq \theta_k + (\pi - \pi^k)^T g^k$, so that imposing $\theta(\pi) \geq \theta^*$ yields the Polyak-Kelly cut $(\pi - \pi^k)^T g^k \geq \theta^* - \theta_k$. In their PKC strategy, Sherali et al. estimated $(\theta^* - \theta_k)$ by $\beta(w_l - \theta_k)$ at iteration k. On the other hand, in our generalized Polyak-Kelly cutting plane strategy (denoted GPKC), we estimate the optimal objective value θ^* by $\hat{\theta}^* \equiv \max\{\theta_{k-i} + \beta(w_l - \theta_{k-i}) : i = 0, \ldots \delta'\}$, for $0 \leq \delta' \leq \delta$. Note that $\delta' = 0$ is used for the PKC approach of Sherali et al., while we employ the maximum estimate of this type produced over the most recent $\delta + 1$ iterations, including the present iteration k. Using this estimate $\hat{\theta}^*$, let us define the following sets induced by the corresponding cutting planes:

$$S_{k-i} \equiv \{\pi \in R^m : (\pi - \pi^{k-i})^T g^{k-i} \geq \hat{\theta}^* - \theta_{k-i}\}, \text{ for } i = 0, 1, \ldots, \text{ and} \tag{3.1a}$$

$$Q_k \equiv S_k \bigcap S_{k-1}. \tag{3.1b}$$

Then, at iteration k, the GPKC strategy performs projections of π^k to obtain $\hat{\pi}^{k+1}$ as follows, for some choice of parameter δ'' satisfying $1 \leq \delta'' \leq \delta$ in case RESET=0 (whence $\delta \geq 1$).

$$\hat{\pi}^{k+1} = \begin{cases} P_{S_k}(\pi^k) & \text{if RESET} = 1 \text{ or } Q_k = \varnothing, \\ P_{Q_k}(\pi^k) & \text{if RESET} = 0 \ \delta'' = 1, \text{and } Q_k \neq \varnothing, \\ \overline{P}_{S_{k-\delta''}} \cdots \overline{P}_{S_{k-2}} P_{Q_k}(\pi^k) & \text{if RESET} = 0, \delta'' \geq 2, \text{ and } Q_k \neq \varnothing, \end{cases} \tag{3.2}$$

where for the case when RESET=0, if $2 \leq \delta'' \leq \delta$, then for $i = 1, \ldots, \delta''$, we have

$$\overline{P}_{S_{k-i}}(\pi) \equiv \begin{cases} P_{S_{k-i}}(\pi) & \text{if } P_{S_{k-i}}(\pi) \in Q_k \\ \pi & \text{otherwise.} \end{cases} \tag{3.3}$$

Again, note that $\delta'' = 1$ is used for PKC of Sherali et al. (2001).

These four subroutines for determining a new iterate π^{k+1} (based on the intermediate iterate $\hat{\pi}^{k+1}$) for use in the aforementioned two variable target value frameworks are summarized as follows.

Subroutine PS

Compute a step-length $\lambda_k = \dfrac{\beta[w_l - \theta_k]}{\left\| g^k \right\|^2}$. Exit the subroutine with $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$, where $\hat{\pi}^{k+1} = \pi^k + \lambda_k g^k$.

Subroutine ADS

If RESET=1, set $d^{k-1} = 0$ and otherwise, determine $\alpha_k = \left\| g^k \right\| / \left\| d^{k-1} \right\|$. Put $d^k = g^k + \alpha_k d^{k-1}$. If $\left\| d^k \right\| = 0$ (practically, $\left\| d^k \right\| \le 10^{-6}$), put $d^k = g^k$. Compute $\lambda_k = \dfrac{\beta[w_l - \theta_k]}{\left\| d^k \right\|^2}$, and exit the subroutine with $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$, where $\hat{\pi}^{k+1} = \pi^k + \lambda_k g^k$.

Subroutine VA

STEP i: If RESET=1, set $d^{k-1} = 0$ and $d^k = g^k$; otherwise, determine $d^k = \alpha g^k + (1-\alpha) d^{k-1}$, where α is a prescribed moving average parameter such that $\alpha \in [\beta, 1]$. Compute $\lambda_k = \dfrac{\beta[w_l - \theta_k]}{\left\| d^k \right\|^2}$, and exit the subroutine with $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$, where $\hat{\pi}^{k+1} = \pi^k + \lambda_k g^k$.

Subroutine GPKC

STEP i: Let $\delta = k - \bar{k}$, where $\bar{k}$ is the most recent iteration at which the variable target value algorithm was reset (i.e. RESET=1), and accordingly, let $\delta' = \min\{\delta, \bar{\delta}'\}$ and $\delta'' = \min\{\delta, \bar{\delta}''\}$, where $\bar{\delta}' \geq 0$ and $\bar{\delta}'' \geq 1$ are prescribed in Step 1 of the variable target value algorithm. Compute

$\hat{\theta}^* = \max\{\theta_{k-i} + \beta(w_l - \theta_{k-i}) : i = 0, \ldots, \delta'\}$.

STEP ii (Projection of π^k onto S_k): Set $\psi_1 = \dfrac{\hat{\theta}^* - \theta_k}{\left\| g^k \right\|^2}$, $\psi_2 = 0$, and $\hat{\pi}^{k+1} = \pi^k + \psi_1 g^k$. If RESET=1, put $\bar{k} = k$ and exit the subroutine with $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$.

STEP iii (Check if $\hat{\pi}^{k+1} \in S_{k-1}$): Compute $\xi_1 = \hat{\theta}^* - \theta_k + (\pi^k)^T g^k$ and $\xi_2 = \hat{\theta}^* - \theta_{k-1} + (\pi^{k-1})^T g^{k-1}$. If $\xi_2 \leq (\hat{\pi}^{k+1})^T g^{k-1}$, go to Step vi.

STEP iv (Projection of π^k onto S_{k-1} and check if this belongs to S_k): Put $\psi_2 = \dfrac{\xi_2 - (\pi^k)^T g^{k-1}}{\left\| g^{k-1} \right\|^2}$, and let $\hat{\pi}^{k+1} = \pi^k + \psi_2 g^{k-1}$. If $\psi_2 > 0$ and $(\hat{\pi}^{k+1})^T g^k \geq \xi_1$, go to Step vi.

STEP v (Check if $Q_k = \varnothing$; otherwise, compute projection of π^k onto Q_k): Compute $\psi_0 = \left\| g^k \right\|^2 \left\| g^{k-1} \right\|^2 - \left((g^{k-1})^T g^k \right)^2$. If $\psi_0 < 10^{-6}$, put $\hat{\pi}^{k+1} = \pi^k + \psi_1 g^k$ and exit the subroutine with $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$. Otherwise, compute

$$\psi_1 = \left[\left\| g^{k-1} \right\|^2 \left(\xi_1 - (\pi^k)^T g^k \right) - \left((g^{k-1})^T g^k \right) \left(\xi_2 - (\pi^k)^T g^{k-1} \right) \right] \Big/ \psi_0 \text{, and}$$

$$\psi_2 = \left[\left\| g^k \right\|^2 \left(\xi_2 - (\pi^k)^T g^{k-1} \right) - \left((g^{k-1})^T g^k \right) \left(\xi_1 - (\pi^k)^T g^k \right) \right] \Big/ \psi_0 .$$

Put $\hat{\pi}^{k+1} = \pi^k + \psi_1 g^k + \psi_2 g^{k-1}$ and proceed to Step vi.

STEP vi (Done with first two cases of (3.2); check for third case): If $\delta'' = 1$, exit the subroutine with $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1})$. Otherwise, put $i = 2$ and proceed to Step vii.

STEP vii (Check if $\hat{\pi}^{k+1} \in S_{k-i}$): Compute $\xi_3 = \hat{\theta}^* - \theta_{k-i} + (\pi^{k-i})^T g^{k-i}$ and let $\psi_3 = \xi_3 - (\hat{\pi}^{k+1})^T g^{k-i}$. If $\psi_3 \leq 0$, go to Step ix. Otherwise, proceed to Step viii.

STEP viii (Projection of $\hat{\pi}^{k+1}$ onto S_{k-i} and check if this belongs to Q_k as per (3.3)): Put

$\overline{\pi}^{k,i} = \hat{\pi}^{k+1} + \dfrac{\psi_3}{\left\| g^{k-i} \right\|^2} g^{k-i}$. If $(\overline{\pi}^{k,i})^T g^k \geq \xi_1$ and $(\overline{\pi}^{k,i})^T g^{k-1} \geq \xi_2$, put

$\hat{\pi}^{k+1} = \overline{\pi}^{k,i}$.

STEP ix (Reiterate for third case of (3.2) as necessary): If $i = \delta''$, exit the subroutine with $\pi^{k+1} = P_{\Pi}(\hat{\pi}^{k+1})$. Otherwise, increment $i \leftarrow i+1$ and return to Step vii.

3.3 Convergence of Algorithms

In this section, we provide convergence analyses for the foregoing algorithmic procedures. Note that if any such algorithm terminates in a finite number of iterations with a zero subgradient, an optimal solution is at hand. Thus, in what follows, we ignore the termination rule predicated by $k_{\max}$, and assume that each algorithm discussed below generates an infinite sequence $\{\pi^k\}$ (i.e., furthermore, the simple case of $\left\| g^k \right\| = 0$ for some k does not hold true).

Note that the convergence argument of the Level Algorithm along with Subroutine PS is already provided by Goffin and Kiwiel (1999), while Sherali et al. (2000) have established the convergence of the VTVM algorithm employing either the pure or the deflected subgradient strategies. Moreover, in Chapter 2, we have proven the convergence of VTVM in concert with Subroutine VA under the condition $0 < \beta \leq \alpha \leq 1$. Therefore, we focus our convergence analyses on the combinations of the Level Algorithm with ADS, or PKC, or VA, and the VTVM algorithm with PKC.

Convergence Analysis for the Level Algorithm

Lemma 3.1

Consider the Level Algorithm along with Subroutine ADS, where the step length parameter β satisfies $0 < \hat{\varepsilon}_1 \leq \beta \leq 1$. Consider an outer iteration l such that $w_l < \theta^*$, where θ^* denotes the optimal objective value. Let $\Lambda(w_l)$ denote the level set $\Lambda(w_l) \equiv \{\pi \in R^m : \theta(\pi) \geq w_l\}$. Then, for any $\tilde{\pi} \in C_l \equiv \Pi \bigcap \Lambda(w_l)$, we have

$$(\pi^k - \tilde{\pi})^T d^{k-1} \le 0, \quad \text{for } k = k(l),\ldots,k(l+1)-1. \tag{3.4}$$

Proof: Consider the most recent iteration $\bar{k} \le k(l)$ for which the Level Algorithm has been reset, i.e., RESET=1. The proof is established by induction for $k = \bar{k},\ldots,k(l+1)-1$. First, note from Subroutine ADS that $d^{\bar{k}-1} = 0$, and hence the inequality in (3.4) is trivial. Now, suppose that the inequality in (3.4) holds true for $\bar{k},\ldots,k$ such that $k < k(l+1)-1$, and consider $k+1$. Then, using $(\pi^{k+1} - \hat{\pi}^{k+1})^T d^k \le 0$ (by the convexity of Π and the projection operation), concavity of θ, induction hypothesis, $w_l \le \theta(\tilde{\pi})$, and $\beta \in (0,1]$, we obtain

$$\begin{aligned}
(\pi^{k+1} - \tilde{\pi})^T d^k &= (\pi^{k+1} - \hat{\pi}^{k+1} + \hat{\pi}^{k+1} - \tilde{\pi})^T d^k \\
&= (\pi^{k+1} - \hat{\pi}^{k+1})^T d^k + (\pi^k + \lambda_k d^k - \tilde{\pi})^T d^k \\
&\le (\pi^k - \tilde{\pi})^T d^k + \lambda_k \left\| d^k \right\|^2 \\
&= (\pi^k - \tilde{\pi})^T (g^k + \alpha_k d^{k-1}) + \beta(w_l - \theta_k) \\
&= (\pi^k - \tilde{\pi})^T g^k + \alpha_k (\pi^k - \tilde{\pi})^T d^{k-1} + \beta(w_l - \theta_k) \\
&\le -(\theta(\tilde{\pi}) - \theta_k) + \beta(w_l - \theta_k) \\
&\le (\beta - 1)(\theta(\tilde{\pi}) - \theta_k) \le 0. \blacksquare
\end{aligned} \tag{3.5}$$

Lemma 3.2

Consider the Level Algorithm along with Subroutine ADS and an outer iteration l described as in Lemma 3.1. Let the sets $\Lambda(w_l)$ and C_l be defined as in Lemma 3.1. Then, for any $\tilde{\pi} \in C_l \equiv \Pi \bigcap \Lambda(w_l)$, we have

$$\left\| \pi^k - \tilde{\pi} \right\|^2 \le \left\| \pi^{k(l)} - \tilde{\pi} \right\|^2 \qquad \text{for } k = k(l),\ldots,k(l+1). \tag{3.6}$$

Proof: The inequality in (3.6) trivially holds true for $k = k(l)$. Hence, suppose that (3.6)

holds true for $k(l),\ldots,k$ such that $k < k(l+1)-1$, and consider $k+1$. Using the convexity of Π and $\tilde{\pi} \in \Pi$, we have

$$\left\|\pi^{k+1}-\tilde{\pi}\right\|^2 \le \left\|\hat{\pi}^{k+1}-\tilde{\pi}\right\|^2 = \left\|\pi^k+\lambda_k d^k-\tilde{\pi}\right\|^2$$
$$= \left\|\pi^k-\tilde{\pi}\right\|^2 + {\lambda_k}^2\left\|d^k\right\|^2 + 2\lambda_k(\pi^k-\tilde{\pi})^T d^k. \qquad (3.7)$$

Using (3.7), the concavity of θ, Lemma 3.1, $w_l \le \theta(\tilde{\pi})$, $0 < \varepsilon_1 \le \beta \le 1$, and the induction hypothesis, we have

$$\left\|\pi^{k+1}-\tilde{\pi}\right\|^2 \le \left\|\pi^k-\tilde{\pi}\right\|^2 + \beta^2\frac{(w_l-\theta_k)^2}{\left\|d^k\right\|^2} + 2\lambda_k(\pi^k-\tilde{\pi})^T(g^k+\alpha_k d^{k-1}) \qquad (3.8a)$$

$$\le \left\|\pi^k-\tilde{\pi}\right\|^2 + \beta^2\frac{(w_l-\theta_k)^2}{\left\|d^k\right\|^2} + 2\lambda_k(\pi^k-\tilde{\pi})^T g^k \qquad (3.8b)$$

$$\le \left\|\pi^k-\tilde{\pi}\right\|^2 + \beta^2\frac{(w_l-\theta_k)^2}{\left\|d^k\right\|^2} - 2\lambda_k(\theta(\tilde{\pi})-\theta_k) \qquad (3.8c)$$

$$\le \left\|\pi^k-\tilde{\pi}\right\|^2 + \beta^2\frac{(w_l-\theta_k)^2}{\left\|d^k\right\|^2} - 2\lambda_k(w_l-\theta_k) \qquad (3.8d)$$

$$= \left\|\pi^k-\tilde{\pi}\right\|^2 + \beta^2\frac{(w_l-\theta_k)^2}{\left\|d^k\right\|^2} - 2\beta\frac{(w_l-\theta_k)^2}{\left\|d^k\right\|^2} \qquad (3.8e)$$

$$\le \left\|\pi^k-\tilde{\pi}\right\|^2 - \hat{\varepsilon}_1\frac{(w_l-\theta_k)^2}{\left\|d^k\right\|^2} \qquad (3.8f)$$

$$\le \left\|\pi^k-\tilde{\pi}\right\|^2 \le \left\|\pi^{k(l)}-\tilde{\pi}\right\|^2. \qquad (3.8g)$$

Finally, for the case of $k+1 = k(l+1)$ in the above induction argument, observe that (3.8a)-(3.8g) hold true if RESET = 0 at iteration $k(l+1)$. Otherwise, note from Step 4 of the Level Algorithm that $\pi^{k(l+1)}$ is reset as the incumbent solution. If the incumbent solution is $\pi^{k(l+1)}$ itself, then again (3.8a)-(3.8g) hold true. Else,

$\pi^{k(l+1)}$ must be one of the iterates $\pi^{k(l)},\ldots,\pi^{k(l+1)-1}$ for which we have already established (3.6). This completes the proof. ■

Lemma 3.3

Consider the Level Algorithm along with Subroutine ADS. Suppose that there exists a finite optimal objective value and that $\left\{\left\|d^k\right\|\right\}$ is bounded. Then, we have $l \to \infty$, and $\sigma_l \to 0$ as $l \to \infty$.

Proof: First, following the proof of Lemma 3.2 in Goffin and Kiwiel (1999), we prove that $l \to \infty$ by contradiction. Suppose that only a finite number of target values are generated, i.e., $l < \infty$. Then, since we must have $\Delta_{k+1} = \Delta_k + \left\|\hat{\pi}^{k+1} - \pi^k\right\| \le R$, and $\{\Delta_k\}$ is monotone increasing for $k \ge k(l)$, we have $\left\|\hat{\pi}^{k+1} - \pi^k\right\| \to 0$. Note that we have $\left\|\hat{\pi}^{k+1} - \pi^k\right\| = \beta(w_l - \theta_k)/\left\|d^k\right\|$. This, together with the boundedness of $\left\{\left\|d^k\right\|\right\}$, implies that $w_l - \theta_k \to 0$. However, since $\theta_k < z_{k(l)} + \sigma_l/2$ for $k \ge k(l)$, $w_l = z_{k(l)} + \sigma_l$, and $\sigma_l > 0$, we have that $w_l - \theta_k > \sigma_l/2 > 0$, which yields a contradiction. Hence, $l \to \infty$. Now, suppose that $\sigma_l \to \sigma_\infty > 0$. Because $\{\sigma_l\}$ is monotone decreasing, being halved when decreased, there exists an L such that $\sigma_l = \sigma_\infty$ for $l \ge L$. This implies that the target value is updated by Step 3 for $l \ge L$. Therefore, we have $z_{k(l+1)} \ge z_{k(l)} + \sigma_\infty/2$ for $l \ge L$, and thereby, $z_{k(l)} \to \infty$ as $l \to \infty$. This contradicts the finiteness of the optimal objective value. ■

Theorem 3.1 (Convergence of the Level Algorithm along with Subroutine ADS)

Consider the Level Algorithm along with Subroutine ADS. Suppose that the initial step-length parameter β satisfies $0 < \hat{\varepsilon}_1 \le \beta \le 1$. Furthermore, suppose that there exists a finite optimal objective value θ^*, and that $\left\{\left\|d^k\right\|\right\}$ is bounded. Then, $\{z_k\} \to \theta^*$.

Proof: This proof extends Goffin and Kiwiel's argument for the simple Level Algorithm.

Assume on the contrary that $z_k \to z_\infty < \theta^*$. Let $\hat{\theta} \in (z_\infty, \theta^*)$ and $\delta \equiv \hat{\theta} - z_\infty$. Then, by Lemma 3.3, there exists an L such that $\sigma_l \le \delta$ for $l \ge L$. Accordingly, we suppose that $l \ge L$. Consider $\tilde{\pi} \in \Pi \bigcap \Lambda(\hat{\theta})$, where $\Lambda(\cdot)$ is the level set defined as in Lemma 3.1. Then, for $k = k(l), \ldots, k(l+1)-1$, we have

$$\theta_k < w_l = z_{k(l)} + \sigma_l < \hat{\theta} - \delta + \sigma_l \le \hat{\theta} \le \theta(\tilde{\pi}),$$

and hence, $\tilde{\pi} \in C_l$ as defined in Lemma 3.1. Let $\tilde{\pi}^k \equiv (1-\mu_k)\tilde{\pi} + \mu_k \pi^k$, where $\mu_k = (\theta(\tilde{\pi}) - w_l)/(\theta(\tilde{\pi}) - \theta_k)$. Then, by the concavity of θ, $\theta(\tilde{\pi}^k) \ge (1-\mu_k)\theta(\tilde{\pi}) + \mu_k \theta(\pi^k) = w_l$, and hence, $\tilde{\pi}^k \in C_l$. From (3.6) in Lemma 3.2 and $\theta(\tilde{\pi}) - \theta_k > \delta$, we have

$$\begin{aligned} \left\| \pi^k - \tilde{\pi}^k \right\| &= (1-\mu_k) \left\| \pi^k - \tilde{\pi} \right\| = \frac{w_l - \theta_k}{\theta(\tilde{\pi}) - \theta_k} \left\| \pi^k - \tilde{\pi} \right\| \\ &\le \left(\frac{\left\| d^k \right\| \left\| \pi^{k(l)} - \tilde{\pi} \right\|}{\delta} \right) \left(\frac{w_l - \theta_k}{\left\| d^k \right\|} \right). \end{aligned} \tag{3.9}$$

Therefore, noting that $\tilde{\pi}^k \in C_l$ and letting $d_{C_l}(\cdot)$ denote the distance from C_l, we have

$$d_{C_l}(\pi^k) \le \left\| \pi^k - \tilde{\pi}^k \right\| \le t_k \frac{w_l - \theta_k}{\left\| d^k \right\|}, \tag{3.10}$$

where $t_k \equiv \left\| d^k \right\| \left\| \pi^{k(l)} - \tilde{\pi} \right\| / \delta$. Let $\pi^k_{C_l} \equiv P_{C_l}(\pi^k)$. Then, from (3.8f), $\pi^k_{C_l} \in C_l$, and (3.10), we have

$$d^2_{C_l}(\pi^{k+1}) \le \left\|\pi^{k+1} - \pi^k_{C_l}\right\|^2 \le \left\|\pi^k - \pi^k_{C_l}\right\|^2 - \hat{\varepsilon}_1 \frac{(w_l - \theta_k)^2}{\left\|d^k\right\|^2}$$

$$\le d^2_{C_l}(\pi^k) - \hat{\varepsilon}_1 \frac{d^2_{C_l}(\pi^k)}{t_k^2} = \left(1 - \frac{\hat{\varepsilon}_1}{t_k^2}\right) d^2_{C_l}(\pi^k)$$

$$\le q_l^2 d^2_{C_l}(\pi^k), \tag{3.11}$$

where $q_l \equiv (1 - \hat{\varepsilon}_1 / t_M^2)^{1/2}$, and where $t_M = M \left\|\pi^{k(l)} - \tilde{\pi}\right\| / \delta \ge t_k \quad \forall k$, taking M such that $\left\|d^k\right\| \le M \quad \forall k$ (by the boundedness of $\left\{\left\|d^k\right\|\right\}$). Thus, $d_{C_l}(\pi^{k+1}) \le q_l d_{C_l}(\pi^k)$. Let us define R_k by $R_k \equiv \{\pi : (\pi - \pi^k)^T d^k \ge \beta(w_l - \theta_k)\}$. Then, we have, for any $\pi_{C_l} \in C_l$, using $\beta \in (0,1]$, the concavity of θ, $d^k = g^k + \alpha_k d^{k-1}$, and Lemma 3.1,

$$\beta(w_l - \theta_k) \le w_l - \theta_k \le \theta(\pi_{C_l}) - \theta_k \le (\pi_{C_l} - \pi^k)^T g^k$$

$$= (\pi_{C_l} - \pi^k)^T d^k - \alpha_{k-1}(\pi_{C_l} - \pi^k)^T d^{k-1}$$

$$\le (\pi_{C_l} - \pi^k)^T d_k, \tag{3.12}$$

and hence, $R_k \supseteq C_l$. This in turn implies that

$$\left\|\hat{\pi}^{k+1} - \pi^k\right\| = \left\|\lambda_k d^k\right\| = \beta \frac{w_l - \theta_k}{\left\|d^k\right\|} = d_{R_k}(\pi^k) \le d_{C_l}(\pi^k), \tag{3.13}$$

where $d_{R_k}(\cdot)$ denotes the distance from R_k. Now, the total length of the directional steps from iteration $k(l)$ to k becomes, using (3.13) and (3.11),

$$\Delta_k = \sum_{j=k(l)}^{k-1} \left\|\hat{\pi}^{j+1} - \pi^j\right\| \le \sum_{j=k(l)}^{k-1} d_{C_l}(\pi^j)$$

$$\leq d_{C_l}(\pi^{k(l)}) \sum_{j=k(l)}^{k-1} q_l^{j-k(l)} < \frac{d_{C_l}(\pi^{k(l)})}{1-q_l}, \ \forall k = k(l),\ldots,k(l+1). \quad (3.14)$$

However, noting $\tilde{\pi}^{k(l)} \in C_l$ and using the convexity of C_l and (3.9), we have

$$d_{C_l}(\pi^{k(l)}) \leq \left\|\pi^{k(l)} - \tilde{\pi}^{k(l)}\right\| \leq \frac{\left\|\pi^{k(l)} - \tilde{\pi}\right\|(w_l - \theta_{k(l)})}{\delta}. \quad (3.15)$$

Note from (3.6) in Lemma 3.2 that we have $\left\|\pi^{k(l+1)} - \tilde{\pi}\right\| \leq \left\|\pi^{k(l)} - \tilde{\pi}\right\|$. Furthermore, we have $\theta_{k(l)} = z_{k(l)}$ whenever the target value is updated either at Step 3 or at Step 4 (while resetting), and hence, $w_l - \theta_{k(l)} = w_l - z_{k(l)} = \sigma_l \ \ \forall l$. Since $\left\{\left\|\pi^{k(l)} - \tilde{\pi}\right\|\right\}$ is monotone decreasing and $\{\sigma_l\} \to 0$ as $l \to \infty$, the final term in (3.15) converges to zero, and so,

$$\left\{d_{C_l}(\pi^{k(l)})\right\} \to 0.$$

Furthermore, since q_l is bounded away from 1, we have from (3.13) and (3.14) that there exists an $\bar{L} \geq L$ such that $\Delta_{k(l+1)} < R$ for $l \geq \bar{L}$. Therefore, $\sigma_{l+1} = \sigma_l > 0$ for $l \geq \bar{L}$, which implies that $\{\sigma_l\} \to \bar{\sigma} > 0$. This contradicts Lemma 3.3. ■

Corollary 3.1.1

Consider the Level Algorithm along with Subroutine VA. Suppose that the initial step-length parameter β satisfies $0 < \hat{\varepsilon}_1 \leq \beta \leq \alpha \leq 1$. Furthermore, suppose that there exists a finite optimal objective value θ^*, and that $\left\{\left\|d^k\right\|\right\}$ is bounded. Then, $\{z_k\} \to \theta^*$.

Proof: Considering Subroutine VA instead of Subroutine ADS, and using that $\beta \leq \alpha$, we can modify the string of inequalities leading to (3.5) as follows:

$$
\begin{aligned}
(\pi^{k+1}-\tilde{\pi})^T d^k &= (\pi^{k+1}-\hat{\pi}^{k+1}+\hat{\pi}^{k+1}-\tilde{\pi})^T d^k \\
&= (\pi^{k+1}-\hat{\pi}^{k+1})^T d^k + (\pi^k+\lambda_k d^k-\tilde{\pi})^T d^k \\
&\le (\pi^k-\tilde{\pi})^T d^k + \lambda_k \left\| d^k \right\|^2 \\
&= (\pi^k-\tilde{\pi})^T (\alpha g^k + (1-\alpha) d^{k-1}) + \beta(w_l-\theta_k) \\
&= \alpha(\pi^k-\tilde{\pi})^T g^k + (1-\alpha)(\pi^k-\tilde{\pi})^T d^{k-1} + \beta(w_l-\theta_k) \\
&\le -\alpha(\theta(\tilde{\pi})-\theta_k) + \beta(w_l-\theta_k) \\
&\le (\beta-\alpha)(\theta(\tilde{\pi})-\theta_k) \le 0 .
\end{aligned}
$$

Since the rest of the proof is identical to that for Lemma 3.1, the inequality (3.4) holds true when Subroutine VA is employed as a direction finding strategy. Using this result, we can rewrite (3.8) as

$$
\begin{aligned}
\left\| \pi^{k+1}-\tilde{\pi} \right\|^2 &\le \left\| \pi^k-\tilde{\pi} \right\|^2 + \beta^2 \frac{(w_l-\theta_k)^2}{\left\| d^k \right\|^2} + 2\lambda_k (\pi^k-\tilde{\pi})^T (\alpha g^k + (1-\alpha) d^{k-1}) \\
&\le \left\| \pi^k-\tilde{\pi} \right\|^2 + \beta^2 \frac{(w_l-\theta_k)^2}{\left\| d^k \right\|^2} + 2\alpha\lambda_k (\pi^k-\tilde{\pi})^T g^k \\
&\le \left\| \pi^k-\tilde{\pi} \right\|^2 + \beta^2 \frac{(w_l-\theta_k)^2}{\left\| d^k \right\|^2} - 2\alpha\lambda_k (\theta(\tilde{\pi})-\theta_k) \\
&\le \left\| \pi^k-\tilde{\pi} \right\|^2 + \beta^2 \frac{(w_l-\theta_k)^2}{\left\| d^k \right\|^2} - 2\alpha\lambda_k (w_l-\theta_k) \\
&= \left\| \pi^k-\tilde{\pi} \right\|^2 + \beta^2 \frac{(w_l-\theta_k)^2}{\left\| d^k \right\|^2} - 2\alpha\beta \frac{(w_l-\theta_k)^2}{\left\| d^k \right\|^2} \\
&\le \left\| \pi^k-\tilde{\pi} \right\|^2 - \hat{\varepsilon}_1 \alpha \frac{(w_l-\theta_k)^2}{\left\| d^k \right\|^2} \\
&\le \left\| \pi^k-\tilde{\pi} \right\|^2 \le \left\| \pi^{k(l)}-\tilde{\pi} \right\|^2 .
\end{aligned}
$$

Thus, (3.6) holds true for Subroutine VA. Moreover, note that the argument in Lemma 3.3 remains the same when Subroutine VA is employed as a direction finding strategy. Finally, using the concavity of θ, $d^k = \alpha g^k + (1-\alpha) d^{k-1}$, (3.4), and $\alpha \in [\beta, 1]$, we get

$$\begin{aligned} w_l - \theta_k &\le \theta(\pi_{C_l}) - \theta_k \le (\pi_{C_l} - \pi^k)^T g^k \\ &= \frac{1}{\alpha}(\pi_{C_l} - \pi^k)^T d^k - \frac{(1-\alpha)}{\alpha}(\pi_{C_l} - \pi^k)^T d^{k-1} \\ &\le \frac{1}{\alpha}(\pi_{C_l} - \pi^k)^T d^k \le \frac{1}{\beta}(\pi_{C_l} - \pi^k)^T d^k, \end{aligned}$$

and hence, we have $\beta(w_l - \theta_k) \le (\pi_{C_l} - \pi^k)^T d^k$, which affirms (3.12) in Theorem 3.1. The required convergence proof then follows similar to that of Theorem 3.1.

■

Lemma 3.4

Consider a variable target value algorithm along with Subroutine GPKC, where the step length parameter β satisfies that $0 < \hat{\varepsilon}_1 \le \beta \le 1$. Consider an outer iteration l such that $w_l < \theta^*$, where θ^* denotes the optimal objective value. Let the sets $\Lambda(w_l)$ and C_l be defined as in Lemma 3.1. Furthermore, let the sets S_k and Q_k be defined as in (3.1a) and (3.1b), respectively, for $k = k(l), \ldots k(l+1) - 1$. Then, the followings hold true.

(a) $C_l \subseteq Q_k \ne \varnothing$ if RESET=0 and $\delta'' = 1$,
$C_l \subseteq Q_k \bigcap S_{k-2} \bigcap \cdots \bigcap S_{k-\delta''} \ne \varnothing$ if RESET=0 and $\delta'' \ge 2$.

(b) $\pi^k \notin S_k$.

(c) $\left\| \hat{\pi}^{k+1} - \tilde{\pi} \right\| \le \left\| P_{Q_k}(\pi^k) - \tilde{\pi} \right\|$ if RESET=0.

(d) $\left\| P_{S_k}(\pi^k) - \pi^k \right\| \ge \dfrac{\beta(w_l - \theta_k)}{\left\| g^k \right\|}$.

Proof: Note from the definition of $\hat{\theta}^*$ that since $\beta \le 1$, we have $\hat{\theta}^* = \theta_k^{\delta'} + \beta(w_l - \theta_k^{\delta'})$, where $\theta_k^{\delta'} = \max_{i=0,\ldots,\delta'} \theta_{k-i}$. For $\tilde{\pi} \in C_l$, using the concavity of θ, $\theta(\tilde{\pi}) \ge w_l$, and $\beta \in (0,1]$, we have $(\tilde{\pi} - \pi^{k-i})^T g^{k-i} \ge \theta(\tilde{\pi}) - \theta_{k-i} \ge w_l - \theta_{k-i}$ $\ge \theta_k^{\delta'} + \beta(w_l - \theta_k^{\delta'}) - \theta_{k-i} = \hat{\theta}^* - \theta_{k-i}$, for $i = 0,\ldots,\delta''$. Hence, $\tilde{\pi} \in S_{k-i}$ for $i = 0,\ldots,\delta''$. This, together with (3.1a), and noting that $C_l \ne \varnothing$ because $w_l < \theta^*$, we have that (a) holds true. Also, since $\theta_k^{\delta'} \ge \theta_k$, $w_l > \theta_k^{\delta'}$, and $\beta \in (0,1]$, we have $0 = (\pi^k - \pi^k)^T g^k < \hat{\theta}^* - \theta_k$. Hence, (b) holds true. Next, note from $Q_k \ne \varnothing$ and RESET=0, if $\delta'' = 1$, then $\hat{\pi}^{k+1} = P_{Q_k}(\pi^k)$ by (3.2), and so, (c) holds true. On the other hand, if $\delta'' \ge 2$, then by (3.2), we have that $\hat{\pi}^{k+1} = \overline{P}_{S_{k-\delta'}} \cdots \overline{P}_{S_{k-2}} P_{Q_k}(\pi^k)$. In this case, let us show that

$$\left\| \hat{\pi}^{k+1} - \pi^k \right\| \equiv \left\| \overline{P}_{S_{k-\delta'}} \overline{P}_{S_{k-\delta'+1}} \cdots \overline{P}_{S_{k-2}} P_{Q_k}(\pi^k) - \tilde{\pi} \right\| \le \left\| P_{Q_k}(\pi^k) - \tilde{\pi} \right\|.$$

For simplicity, assume that $\delta'' \ge 3$ (the case of $\delta'' = 2$ follows identically from below). Note that if $P_{S_{k-i}}\{\overline{P}_{S_{k-\delta'+1}} \cdots \overline{P}_{S_{k-2}} P_{Q_k}(\pi^k)\} \notin Q_k$, we have from (3.3) that $\overline{P}_{S_{k-\delta'}}\{\overline{P}_{S_{k-\delta'+1}} \cdots \overline{P}_{S_{k-2}} P_{Q_k}(\pi^k)\} = \overline{P}_{S_{k-\delta'+1}} \cdots \overline{P}_{S_{k-2}} P_{Q_k}(\pi^k)$; else, we have $\overline{P}_{S_{k-\delta'}}\{\overline{P}_{S_{k-\delta'+1}} \cdots \overline{P}_{S_{k-2}} P_{Q_k}(\pi^k)\} = P_{S_{k-\delta'} \cap Q_k}\{\overline{P}_{S_{k-\delta'+1}} \cdots \overline{P}_{S_{k-2}} P_{Q_k}(\pi^k)\}$. For either case, we have $\left\| \overline{P}_{S_{k-\delta'}}\{\overline{P}_{S_{k-\delta'+1}} \cdots \overline{P}_{S_{k-\delta'}} P_{Q_k}(\pi^k)\} - \tilde{\pi} \right\| \le \left\| \overline{P}_{S_{k-\delta'+1}} \cdots \overline{P}_{S_{k-2}} P_{Q_k}(\pi^k) - \tilde{\pi} \right\|$ (using the convexity of $S_{k-\delta'}$ and $\tilde{\pi} \in S_{k-\delta'}$ from the proof of part (a) for the latter case). Continuing in this fashion, we have that (c) holds true. Finally, since $\beta \in (0,1]$, we have $\left\| P_{S_k}(\pi^k) - \pi^k \right\| = \dfrac{\hat{\theta}^* - \theta_k}{\left\| g^k \right\|} = \dfrac{\theta_k^{\delta'} + \beta(w_l - \theta_k^{\delta'}) - \theta_k}{\left\| g^k \right\|}$

$= \dfrac{\beta(w_l - \theta_k) + (1-\beta)(\theta_k^{\delta'} - \theta_k)}{\left\| g^k \right\|} \ge \dfrac{\beta(w_l - \theta_k)}{\left\| g^k \right\|}$. This completes the proof. ■

Lemma 3.5

Consider the Level Algorithm along with Subroutine GPKC, where the step length parameter β satisfies $0 < \hat{\varepsilon}_1 \le \beta \le 1$, and where the cutting plane parameter $\bar{\delta}'' = 1$. Consider an outer iteration l such that $w_l < \theta^*$, where θ^* denotes the optimal objective value. Let the sets $\Lambda(w_l)$ and C_l be defined as in Lemma 3.1. Then, for any $\tilde{\pi} \in C_l$, we have

$$\left\| \pi^k - \tilde{\pi} \right\|^2 \le \left\| \pi^{k(l)} - \tilde{\pi} \right\|^2 \qquad \text{for } k = k(l), \ldots, k(l+1). \tag{3.16}$$

Proof: The proof is similar to that for Lemma 3.2. The inequality in (3.16) trivially holds true for $k = k(l)$. Hence, suppose that (3.16) holds true for $k(l), \ldots, k$ such that $k < k(l+1) - 1$, and consider $k+1$. Note that using the concavity of θ, $\theta(\tilde{\pi}) \ge w_l$, and $0 < \beta \le 1$, we have

$$(\tilde{\pi} - \pi^k)^T g^k \ge \theta(\tilde{\pi}) - \theta_k \ge w_l - \theta_k \ge \beta(w_l - \theta_k). \tag{3.17}$$

If $k = k(l)$ and RESET=1, then, noting that $\hat{\theta}^* = \theta_k + \beta(w_l - \theta_k)$ and using $\pi^{k+1} = P_\Pi(\hat{\pi}^{k+1}) = P_\Pi(P_{\hat{S}_k}(\pi^k))$, the convexity of Π, $\tilde{\pi} \in \Pi$, $\lambda_k \ge 0$, (3.17), and $0 < \hat{\varepsilon}_1 \le \beta \le 1$, we have

$$\left\| \pi^{k+1} - \tilde{\pi} \right\|^2 \le \left\| \hat{\pi}^{k+1} - \tilde{\pi} \right\|^2 = \left\| \pi^k + \frac{\beta(w_l - \theta_k)}{\left\| g^k \right\|^2} g^k - \tilde{\pi} \right\|^2 \tag{3.18a}$$

$$= \left\| \pi^k - \tilde{\pi} \right\|^2 + \frac{\beta^2 (w_l - \theta_k)^2}{\left\| g^k \right\|^2} + 2\frac{\beta(w_l - \theta_k)}{\left\| g^k \right\|^2} (\pi^k - \tilde{\pi})^T g^k \tag{3.18b}$$

$$\le \left\| \pi^k - \tilde{\pi} \right\|^2 + \frac{\beta^2 (w_l - \theta_k)^2}{\left\| g^k \right\|^2} - 2\frac{\beta^2 (w_l - \theta_k)^2}{\left\| g^k \right\|^2} \tag{3.18c}$$

$$= \left\| \pi^k - \tilde{\pi} \right\|^2 - \beta^2 \frac{(w_l - \theta_k)^2}{\left\| g^k \right\|^2} \tag{3.18d}$$

$$\leq \left\| \pi^k - \tilde{\pi} \right\|^2 - \hat{\varepsilon}_1^2 \frac{(w_l - \theta_k)^2}{\left\| g^k \right\|^2} \tag{3.18e}$$

$$\leq \left\| \pi^k - \tilde{\pi} \right\|^2 \leq \left\| \pi^{k(l)} - \tilde{\pi} \right\|^2 . \tag{3.18f}$$

Otherwise, we have RESET=0 and from (a) of Lemma 3.4, we have that $Q_k \neq \varnothing$. Using $\bar{\delta}'' = 1$, (3.2), the convexity of Π, the convexity of Q_k together with (a) of Lemma 3.4, $S_k \supseteq Q_k$ together with (b) of Lemma 3.4, (d) of Lemma 3.4, $0 < \hat{\varepsilon}_1 \leq \beta \leq 1$, and the induction hypothesis, we have

$$\left\| \pi^{k+1} - \tilde{\pi} \right\|^2 \leq \left\| \hat{\pi}^{k+1} - \tilde{\pi} \right\|^2 = \left\| P_{Q_k}(\pi^k) - \tilde{\pi} \right\|^2 = \left\| P_{Q_k}(\pi^k) - \pi^k + \pi^k - \tilde{\pi} \right\|^2 \tag{3.19a}$$

$$= \left\| P_{Q_k}(\pi^k) - \pi^k \right\|^2 + \left\| \pi^k - \tilde{\pi} \right\|^2 + 2\left(P_{Q_k}(\pi^k) - \pi^k \right)^T (\pi^k - \tilde{\pi}) \tag{3.19b}$$

$$= \left\| \pi^k - \tilde{\pi} \right\|^2 - \left\| P_{Q_k}(\pi^k) - \pi^k \right\|^2 - 2\left(\pi^k - P_{Q_k}(\pi^k) \right)^T \left(P_{Q_k}(\pi^k) - \tilde{\pi} \right) \tag{3.19c}$$

$$\leq \left\| \pi^k - \tilde{\pi} \right\|^2 - \left\| P_{Q_k}(\pi^k) - \pi^k \right\|^2 \tag{3.19d}$$

$$\leq \left\| \pi^k - \tilde{\pi} \right\|^2 - \left\| P_{S_k}(\pi^k) - \pi^k \right\|^2 \tag{3.19e}$$

$$\leq \left\| \pi^k - \tilde{\pi} \right\|^2 - \beta^2 \frac{(w_l - \theta_k)^2}{\left\| g^k \right\|^2} \tag{3.19f}$$

$$\leq \left\| \pi^k - \tilde{\pi} \right\|^2 - \hat{\varepsilon}_1^2 \frac{(w_l - \theta_k)^2}{\left\| g^k \right\|^2} \tag{3.19g}$$

$$\leq \left\| \pi^k - \tilde{\pi} \right\|^2 \leq \left\| \pi^{k(l)} - \tilde{\pi} \right\|^2 . \tag{3.19h}$$

Using (3.18) and (3.19), the remainder of the proof is identical to that for Lemma 3.2. ■

Lemma 3.6

Consider the Level Algorithm along with Subroutine GPKC, where the step length parameter β satisfies $0 < \hat{\varepsilon}_1 \le \beta \le 1$, and where $\bar{\delta}'' = 1$. Suppose that there exists a finite optimal objective value and that $\{\|g^k\|\}$ is bounded. Then, we have $l \to \infty$, and $\sigma_l \to 0$ as $l \to \infty$.

Proof: Let S_k and Q_k denote the sets as defined in (3.1a) and (3.1b), respectively. Then, for $k \ge k(l)+1$, since RESET=0, noting from (3.2) that $\hat{\pi}^{k+1} = P_{Q_k}(\pi^k)$, and using $Q_k \subseteq S_k$ and (d) of Lemma 3.4, we have $\|\hat{\pi}^{k+1} - \pi^k\| = \|P_{Q_k}(\pi^k) - \pi^k\| \ge \|P_{S_k}(\pi^k) - \pi^k\| \ge \beta(w_l - \theta_k)/\|g^k\|$, and using the boundedness of $\{\|g^k\|\}$, the proof is identical to that for Lemma 3.3. ■

Theorem 3.2

Consider the Level Algorithm along with Subroutine GPKC, where the step length parameter β satisfies $0 < \hat{\varepsilon}_1 \le \beta \le 1$, and where $\bar{\delta}'' = 1$. Furthermore, suppose that there exists a finite optimal objective value θ^* and that $\|g^k\|$ is bounded. Then, $\{z_k\} \to \theta^*$.

Proof: Assume on the contrary that $z_k \to z_\infty < \theta^*$. Then, the proof is identical to that for Theorem 3.1 until the point where from (3.16) in Lemma 3.5 and $\theta(\tilde{\pi}) - \theta_k > \delta$, we revise (3.9) as

$$\|\pi^k - \tilde{\pi}^k\| = (1-\mu_k)\|\pi^k - \tilde{\pi}\| = \frac{w_l - \theta_k}{\theta(\tilde{\pi}) - \theta_k}\|\pi^k - \tilde{\pi}\|$$

$$\le \left(\frac{\|g^k\|\|\pi^{k(l)} - \tilde{\pi}\|}{\delta}\right)\left(\frac{w_l - \theta_k}{\|g^k\|}\right).$$

Hence, in lieu of (3.10), we now get

$$d_{C_l}(\pi^k) \le \left\|\pi^k - \tilde{\pi}^k\right\| \le t_k \frac{w_l - \theta_k}{\left\|g^k\right\|}, \tag{3.20}$$

where $t_k \equiv \left\|g^k\right\| \left\|\pi^{k(l)} - \tilde{\pi}\right\| / \delta$. Let $\pi^k_{C_l} \equiv P_{C_l}(\pi^k)$. Then, from (3.18e), (3.19g), $\pi^k_{C_l} \in C_l$, and (3.20), we modify (3.11) as

$$\begin{aligned} d^2_{C_l}(\pi^{k+1}) &\le \left\|\pi^{k+1} - \pi^k_{C_l}\right\|^2 \le \left\|\pi^k - \pi^k_{C_l}\right\|^2 - \hat{\varepsilon}_1^2 \frac{(w_l - \theta_k)^2}{\left\|g^k\right\|^2} \\ &\le d^2_{C_l}(\pi^k) - \hat{\varepsilon}_1^2 \frac{d^2_{C_l}(\pi^k)}{t_k^2} = \left(1 - \frac{\hat{\varepsilon}_1^2}{t_k^2}\right) d^2_{C_l}(\pi^k) \\ &\le q_l^2 d^2_{C_l}(\pi^k), \end{aligned}$$

where $q_l \equiv (1 - \hat{\varepsilon}_1^2 / t_M^2)^{1/2}$, and where $t_M = M \left\|\pi^{k(l)} - \tilde{\pi}\right\| / \delta \ge t_k \quad \forall k$, taking M such that $\left\|g^k\right\| \le M \quad \forall k$ (by the boundedness of $\left\{\left\|g^k\right\|\right\}$). Thus, $d_{C_l}(\pi^{k+1}) \le q_l d_{C_l}(\pi^k)$. Let S_k and Q_k denote the sets as defined in (3.1a) and (3.1b), respectively. If RESET=1 at iteration k, noting that $C_l \subseteq S_k$, we have $\left\|\hat{\pi}^{k+1} - \pi^k\right\| = d_{S_k}(\pi^k) \le d_{C_l}(\pi^k)$. Otherwise, we have $C_l \subseteq Q_k$ from (a) of Lemma 3.4, and hence, we get $\left\|\hat{\pi}^{k+1} - \pi^k\right\| = d_{Q_k}(\pi^k) \le d_{C_l}(\pi^k)$. In either case, similar to (3.12), we get

$$\left\|\hat{\pi}^{k+1} - \pi^k\right\| \le d_{C_l}(\pi^k).$$

Except for using Lemma 3.5 and Lemma 3.6 instead of Lemma 3.2 and Lemma 3.3, respectively, the remainder of the proof is identical to that for Theorem 3.1. ■

Convergence Analysis for the VTVM algorithm along with Subroutine PKC

Note that the VTVM algorithm restarts at the incumbent solution whenever the target value is modified via Step 6. However, for convergence purposes as recommended in Sherali et al. (2000), we assume that no such restarts are performed after some finite number of iterations. Then, the convergence analysis of the VTVM algorithm along with Subroutine GPKC can be established as follows.

Lemma 3.7

Consider the VTVM algorithm along with Subroutine GPKC, where no restarts are performed after some iteration K. Suppose that the initial step-length parameter β satisfies $0 < \varepsilon_1 \le \beta \le 1$ and $\delta'' \ge 2$. Let us denote the target value at any iteration k by $\hat{w}_k$. Suppose that there exist values $\underline{w}$ and $\overline{w}$ such that

$$\theta_k \le \underline{w} < \hat{w}_k < \overline{w} \le \theta^*, \qquad \forall k \ge K .$$

Then we have $\{\hat{w}_k\} \to \underline{w}$ and $\{\theta_k\} \to \underline{w}$.

Proof: Let π^* be a dual optimum. Then, noting that $\pi^* \in C_l$, and replacing $\tilde{\pi}$ by π^* and w_l by $\hat{w}_k$ in (3.18a)-(3.18e) if RESET=1, and in (3.19a)-(3.19g) together with noting (c) of Lemma 3.4 otherwise, we have

$$\left\| \pi^{k+1} - \pi^* \right\|^2 \le \left\| \pi^k - \pi^* \right\|^2 - \hat{\varepsilon}_1^2 \frac{(\hat{w}_k - \theta_k)^2}{\left\| g^k \right\|^2} .$$

Therefore, $\left\{ \left\| \pi^k - \pi^* \right\|^2 \right\}$ is monotone decreasing and since it is bounded below, it is convergent. This implies that $\{\hat{w}_k - \theta_k\} \to 0$, and hence, $\{\hat{w}_k\} \to \underline{w}$ and $\{\theta_k\} \to \underline{w}$. ■

Theorem 3.3

Consider the VTVM algorithm along with Subroutine GPKC, where no restarts at the incumbent solution are performed after a finite number of iterations. Suppose that the initial step-length parameter β satisfies $0 < \varepsilon_1 \leq \beta \leq 1$ and $\delta'' \geq 2$. Furthermore, suppose that there exists a finite optimal objective value θ^*, and that $\{\|g^k\|\}$ is bounded. Then, $\{z_k\} \to \theta^*$.

Proof: Note that Lemma 2 and Lemma 3 of Sherali et al. (2000) hold true regardless of the direction finding strategy and the condition imposed on β. Hence, assuming that $\{z_k\} \to \bar{z} < \theta^*$, these lemmas lead to a contradiction of the assertion $\{\theta_k\} \to \underline{w}$ of Lemma 3.7 as in the proof for Theorem 1 of Sherali et al. (2000). Therefore, we have $\{z_k\} \to \theta^*$. ■

3.4 Modifications of the Variable Target Value Algorithms

In this section, we discuss some potential deficiencies of the two variable target value algorithms presented in Section 2, which if alleviated, could result in improving their performance. Motivated by these insights, we suggest some modifications of these methods. Furthermore, we employ an additional local search procedure in order to strengthen the performance of these algorithms.

Modification of the VTVM Algorithm

Notice that target values are increased via $w_{l+1} = z_k + \varepsilon_l + \eta\Delta$ at Step 5 of the VTVM algorithm. Suppose that $w_l - z_{k-1}$ is small. Then, ε_l is small because it is a fraction of $w_l - z_{k-1}$, i.e., $\varepsilon_l = \sigma(w_l - z_{k-1})$, where $\sigma \in (0, 1/3]$. Furthermore, Δ is relatively small because small target values yield small step-lengths. Therefore, $\varepsilon_l + \eta\Delta = w_{l+1} - z_k$ becomes small and, in turn, this causes the next target value not to increase substantially. A simple way to remedy this crawling phenomenon is to increase the algorithmic parameter η, which is a factor used to multiply the amount of

improvement in the objective value since the last update of the target value. Note that the value of η recommended by Sherali et al. (2000) is 0.75, and thus, only 75% of the most recent improvement contributes to increasing the target value. Therefore, we could allow $\eta > 1$ to promote larger increments in the target values during early iterations. However, for convergence purposes, we can revert the value of η to the interval $(0,1]$ toward the later stages of the algorithm. One approach is to adjust the value of η by an acceleration factor of $f_a > 1$, say $f_a = 2$, in order to make the target value more adaptive whenever the target value is updated. Hence, starting with $\eta = 0.75$, we increase η according to

$$\eta \leftarrow f_a \eta,$$

whenever the target value is increased, and we update η according to

$$\eta \leftarrow \max\{\eta / f_a, 0.75\},$$

whenever the target value is decreased. For the sake of theoretical convergence, we fix η at 0.75 after 1500 iterations.

Modification of the Level Algorithm

Notice that target values are decreased by reducing σ_{l+1} to $\sigma_l / 2$ at Step 4. However, as $w_l - z_{k(l)}$ decreases, it is very difficult for Δ_k to exceed the threshold value R because of the smaller resulting step-lengths. That is, the number of iterations taken to decrease the current target value tends to be doubled as compared with that taken for the previous decrement. However, if we ascribe too small a value to R to avoid this, then the target values could get significantly reduced even during early iterations. To remedy this problem, instead of using a fixed prescribed value of R, we can vary it as we proceed into later iterations, that is, we can start with $R_1 = 1.0 \times 10^4$ as recommended by Goffin and Kiwiel (1999), and then reduce it by

$$R_{l+1} \leftarrow \max\{rR_l, \overline{R} = 10\}, \tag{3.21}$$

where $0 < r < 1$, whenever σ_l decreases. The motivation for using the threshold $\overline{R} = 10$ is that if $\{R_l\}$ converges to zero, the convergence argument of Theorem 3.1 no longer holds true because the inequality $\Delta_{k(l+1)} < R_l$ for $l \geq \overline{L}$, in the second last sentence of the proof of Theorem 3.1, is not valid for an arbitrarily small value of R_l. Therefore, we introduce a threshold value $\overline{R} = 10$ as in (3.21) above.

A Further Enhancement of the Variable Target Value Methods

Note that although the NDO problem involves the maximization of a concave objective function over a convex feasible region, a direct employment of differentiable optimization algorithms does not work because a subgradient does not guarantee an ascent direction. However, if an ascent direction is given, a line-search could be beneficially utilized. With this motivation, Sherali and Myers (1988) locally explored for a better solution using a pattern-search whenever an improvement in the objective value is attained. Using the local exploration above, Sherali and Myers (1988) consistently obtained better results for most of their test problems than they did without using the pattern-search. However, if improvements are persistent while the step-length is small, the foregoing local pattern-search requires a significant amount of additional effort.

In lieu of a local pattern-search strategy as above, we could employ a suitable variation of the quadratic-fit line-search. Note that because it is possible that a trial point $\pi^k + \tilde{\lambda}(\pi^k - \pi^{k-1})$ is not feasible to LD, a direct application of the quadratic-fit line-search can fail to find a three-point-pattern. Therefore, we could replace $\theta(\pi^k + \tilde{\lambda}(\pi^k - \pi^{k-1}))$ by the objective value at the projected point $P_\Pi(\pi^k + \tilde{\lambda}(\pi^k - \pi^{k-1}))$ while performing the quadratic-fit line-search along the direction $\pi^k - \pi^{k-1}$. Because the evaluation of $\theta(\cdot)$ could be expensive, an exact line-search is not advisable. Instead, we suggest a single *projected* quadratic-fit line-search, in which a quadratic curve is fitted only once in addition to using the aforementioned projections. (See Bazaraa et al. (1993) for details of the standard quadratic-fit line-search.) This line-

search routine is invoked whenever an improvement in the objective value is detected, and is given as follows.

Projected Quadratic-Fit Line-Search

STEP 1: Initialize $\tilde{\lambda}_1 = 0$, $\tilde{\lambda}_2 = 1$, and $\tilde{\lambda}_3 = 2$. Set the search direction $\tilde{d} = \pi^k - \pi^{k-1}$.

STEP 2: If $\tilde{\lambda}_1$, $\tilde{\lambda}_2$, and $\tilde{\lambda}_3$ satisfy that $\theta(P_\Pi(\pi^{k-1} + \tilde{\lambda}_2\tilde{d})) \geq \theta(P_\Pi(\pi^{k-1} + \tilde{\lambda}_3\tilde{d}))$, go to Step 3. Otherwise, update $\tilde{\lambda}_1 \leftarrow \tilde{\lambda}_2$, $\tilde{\lambda}_2 \leftarrow \tilde{\lambda}_3$, $\tilde{\lambda}_3 \leftarrow 2\tilde{\lambda}_3$, and repeat Step 2.

STEP 3: Calculate $\tilde{\lambda} \equiv \dfrac{b_{23}\tilde{\theta}_1 + b_{31}\tilde{\theta}_2 + b_{12}\tilde{\theta}_3}{2(a_{23}\tilde{\theta}_1 + a_{31}\tilde{\theta}_2 + a_{12}\tilde{\theta}_3)}$, where $a_{ij} = \tilde{\lambda}_i - \tilde{\lambda}_j$, $b_{ij} = \tilde{\lambda}_i^{\,2} - \tilde{\lambda}_j^{\,2}$, and $\tilde{\theta}_i = \theta(P_\Pi(\pi^{k-1} + \tilde{\lambda}_i\tilde{d}))$, $\forall i, j$. If $\theta(P_\Pi(\pi^{k-1} + \tilde{\lambda}\tilde{d})) > \theta(P_\Pi(\pi^{k-1} + \tilde{\lambda}_2\tilde{d}))$, return $\pi^k = P_\Pi(\pi^{k-1} + \tilde{\lambda}\tilde{d})$. Otherwise, return $\pi^k = P_\Pi(\pi^{k-1} + \tilde{\lambda}_2\tilde{d})$.

3.5 Computational Study

In this section, we conduct a comprehensive computational study, in which we implement all algorithmic procedures described in Section 3.2 and their modifications proposed in Section 3.4 to solve dual formulations of four types of test problems used in Chapter 2. Furthermore, we added two set partitioning problems obtained from the OR-Library of the Imperial College Management School (Beasley (1990), Hoffman and Padberg (1993)), and summarized in Table 3.1. Again, all runs, including those of CPLEX, Version 8.1 (which was used for benchmarking purposes), have been made on a Pentium III 667Mhz computer.

The original variable target value procedures described in Section 3.2 as well as their modifications proposed in Section 3.4 were composed with the subroutines PS, VA, ADS, and GPKC for solving the above test problems. For Subroutine GPKC, we tested various combinations of parameter values $\bar{\delta}' = 0, 1, 2, 3$ and $\bar{\delta}'' = 1, 2, 3, 4$, denoted by GPKC($\bar{\delta}', \bar{\delta}''$). (Recall that the PKC strategy of Sherali et al. (2001) corresponds to GPKC(0,1).) Commencing with an initial iterate given by the origin of the corresponding dual space, all algorithms for all problems were run for 2000 iterations. We report the scale-independent statistic POR (percentage optimality ratio) as defined in Chapter 2.

Table 3.1 Summary of Set Partitioning Problems.

Problem Type	Problem	Number of variables	Number of constraints	CPLEX solution	
				Optimal value	CPU time (sec)
Set Partitioning	SPPNW16	148633	139	1181590	32.44
	SPPUS01	1053137	145	9963.07	241.93

Tables 3.2 and 3.3 display the results obtained for the average POR values for each method and class of problems.

Considering the original target value procedures, the VTVM algorithm consistently outperformed the Level Algorithm when applied to MC, TR, LP, and SC (Table 3.2). The smallest difference between the Level Algorithm and VTVM is the average POR value of 3.07%, which occurred when Subroutine VA was employed for LP. In particular, the Level Algorithm displayed two-digit POR values for most of the MC, LP, and SC problems. This slow convergence is due to the weakness of the Level Algorithm as described in Section 3.4. However, when applied to SP, the VTVM algorithm suffered from a severe crawling phenomenon, while the Level Algorithm attained the POR values of less than 1%. For this class of problems, we observed that the initial target values prescribed by VTVM were relatively small, and these values could not be adequately increased rapidly enough because of the potential drawback of VTVM as described in Section 3.4.

The average POR values for the modified variable target value procedures are displayed in Table 3.3. Note that using the modifications as proposed in Section 3.4, the POR values for the VTVM algorithm are drastically decreased by at least 93.2% when applied to the SP problems. Furthermore, this modification of VTVM entailed no significant negative impact when solving the other types of test problems. For example, the largest increment in the average POR values was 0.93% when the modified VTVM algorithm was combined with Subroutine ADS for solving the TR problems. The performance of the modified Level Algorithm was also significantly improved when applied to MC, TR, LP, and SC, while being worsened for the SP problems. Generally, the modifications revealed improved performances for each variable target value

Table 3.2 Average POR Values for Original Target Value Frameworks.

Variable Target Value Framework	Subroutine	Problem Type					Overall
		MC	TR	LP	SC	SP	
VTVM	PS	0	1.16	0.17	0.55	93.40	3.66
	VA	0	2.09	0.14	0.39	93.40	3.71
	ADS	0	3.34	0.13	0.45	96.41	3.93
	GPKC(0,1)	0	1.08	0.13	0.33	93.40	3.61
	GPKC(0,2)	0	1.08	0.13	0.36	93.40	3.61
	GPKC(0,3)	0	1.08	0.13	0.48	93.40	3.61
	GPKC(0,4)	0	1.05	0.14	0.37	93.40	3.61
	GPKC(1,1)	0	1.04	0.13	0.38	93.40	3.61
	GPKC(1,2)	0	1.04	0.13	0.37	93.40	3.61
	GPKC(1,3)	0	1.05	0.13	0.37	93.40	3.61
	GPKC(1,4)	0	1.10	0.14	0.31	93.40	3.61
	GPKC(2,1)	0	1.16	0.13	0.36	93.40	3.62
	GPKC(2,2)	0	1.16	0.13	0.33	93.40	3.62
	GPKC(2,3)	0	1.16	0.13	0.40	93.40	3.62
	GPKC(2,4)	0	1.16	0.13	0.36	93.40	3.62
	GPKC(3,1)	0	1.01	0.13	0.30	93.40	3.60
	GPKC(3,2)	0	1.02	0.13	0.35	93.40	3.60
	GPKC(3,3)	0	0.97	0.13	0.37	93.40	3.60
	GPKC(3,4)	0	1.05	0.13	0.34	93.40	3.61
Level Algorithm	PS	50.23	21.66	4.58	34.25	0.12	11.75
	VA	15.10	23.98	3.21	36.38	0.08	7.88
	ADS	5.54	34.77	3.58	47.50	0.21	8.87
	GPKC(0,1)	21.55	22.34	4.46	35.41	0.08	9.17
	GPKC(0,2)	21.55	22.32	4.46	33.52	0.08	9.06
	GPKC(0,3)	21.55	22.26	4.46	33.36	0.07	9.05
	GPKC(0,4)	21.55	22.39	4.46	33.08	0.08	9.05
	GPKC(1,1)	23.31	22.48	4.32	30.67	0.10	8.98
	GPKC(1,2)	23.31	22.48	4.32	31.06	0.10	9.01
	GPKC(1,3)	23.31	22.77	4.32	34.99	0.11	9.25
	GPKC(1,4)	23.31	22.96	4.27	35.80	0.10	9.27
	GPKC(2,1)	22.22	22.61	4.61	31.60	0.09	9.16
	GPKC(2,2)	22.22	22.56	4.61	32.92	0.10	9.23
	GPKC(2,3)	22.22	22.64	4.61	32.50	0.11	9.21
	GPKC(2,4)	22.22	22.43	4.63	35.75	0.09	9.38
	GPKC(3,1)	21.06	22.50	4.63	32.26	0.08	9.09
	GPKC(3,2)	21.06	22.28	4.63	31.88	0.06	9.05
	GPKC(3,3)	21.06	22.20	4.63	32.98	0.10	9.10
	GPKC(3,4)	21.06	22.16	4.61	36.74	0.08	9.29

framework. Among all compositions of the variable target value procedures and the direction finding and step-length strategies, the modified VTVM algorithm along with

Table 3.3 Average POR Values for Modified Target Value Frameworks.

Variable Target Value Framework	Subroutine	Problem Type					Overall
		MC	TR	LP	SC	SP	
VTVM	PS	0	1.19	0.21	0.38	0.10	0.29
	VA	0	2.12	0.14	0.61	0.18	0.33
	ADS	0	4.27	0.08	0.10	0.03	0.45
	GPKC(0,1)	0	1.28	0.17	0.41	0.04	0.26
	GPKC(0,2)	0	1.10	0.17	0.32	0.04	0.24
	GPKC(0,3)	0	0.95	0.17	0.32	0.04	0.23
	GPKC(0,4)	0	0.74	0.15	0.26	0.05	0.19
	GPKC(1,1)	0	1.19	0.17	0.34	0.03	0.25
	GPKC(1,2)	0	1.15	0.17	0.35	0.09	0.25
	GPKC(1,3)	0	1.14	0.17	0.31	0.07	0.24
	GPKC(1,4)	0	0.82	0.15	0.31	0.07	0.20
	GPKC(2,1)	0	1.24	0.17	0.34	0.04	0.26
	GPKC(2,2)	0	0.99	0.17	0.33	0.07	0.23
	GPKC(2,3)	0	1.06	0.17	0.34	0.09	0.24
	GPKC(2,4)	0	0.88	0.16	0.32	0.08	0.21
	GPKC(3,1)	0	1.22	0.18	0.35	0.06	0.26
	GPKC(3,2)	0	1.10	0.18	0.35	0.04	0.25
	GPKC(3,3)	0	0.97	0.18	0.31	0.09	0.24
	GPKC(3,4)	0	1.00	0.16	0.28	0.04	0.22
Level Algorithm	PS	0.029	9.50	1.69	0.72	10.48	2.51
	VA	0.003	6.23	1.12	0.62	7.06	1.67
	ADS	2.52	7.98	0.28	0.13	0.62	1.19
	GPKC(0,1)	0.003	9.09	1.14	0.67	5.36	1.89
	GPKC(0,2)	0.003	8.94	1.13	0.71	4.81	1.85
	GPKC(0,3)	0.003	8.89	1.13	0.63	4.20	1.82
	GPKC(0,4)	0.003	8.87	1.05	0.54	4.88	1.78
	GPKC(1,1)	0	9.11	1.14	0.66	5.03	1.87
	GPKC(1,2)	0	8.93	1.14	0.70	4.35	1.84
	GPKC(1,3)	0	8.96	1.12	0.59	8.65	1.98
	GPKC(1,4)	0	8.99	1.04	0.52	5.44	1.80
	GPKC(2,1)	0.013	9.09	1.14	0.65	4.72	1.86
	GPKC(2,2)	0.013	8.97	1.13	0.66	4.21	1.83
	GPKC(2,3)	0.013	8.88	1.13	0.61	4.51	1.83
	GPKC(2,4)	0.013	9.03	1.04	0.62	5.04	1.80
	GPKC(3,1)	0.003	9.13	1.13	0.68	4.74	1.86
	GPKC(3,2)	0.003	8.94	1.13	0.66	6.66	1.92
	GPKC(3,3)	0.003	8.87	1.15	0.57	5.25	1.86
	GPKC(3,4)	0.003	9.01	1.03	0.56	3.04	1.71

Subroutine GPKC having $\bar{\delta}' = 0$ and $\bar{\delta}'' = 4$ revealed the best overall average POR value of 0.19%. However, the modified Level Algorithm worked best when combined with the

ADS strategy. We also comment that ADS revealed the best average POR values for MC,LP, SC, and SP, while it did not work well for the class of transportation problems (average POR value of 4.27%).

To see the effect of the enhancements proposed in Section 3.4, we also implemented the modified VTVM algorithm in concert with GPKC(0, 4) without the projected quadratic-fit line-search. The average POR values for this procedure for the classes of test problems MC, TR, LP, SC, and SP were 0%, 1,09%, 1.04%, 0.42%, and 2.79%, respectively. Note that the projected quadratic-fit line-search helped reduce the average POR values for TR, LP, SC, and SP by 0.35%, 0.89%, 0.16%, and 2.74%, respectively. Overall, we therefore recommend using the projected quadratic-fit line-search procedure.

Noting that the heuristic subgradient method of Held et al. (1974) is widely used in practice, we also ran this procedure in order to compare its performance against the modified VTVM algorithm in combination with GPKC(0,4), which revealed the best average POR value. In our runs of Held et al., we experimented with three upper bounds on the optimal objective value: θ^*, $\theta^* + (\theta^* - \theta_1) \times 10\%$, and $\theta^* + (\theta^* - \theta_1) \times 20\%$. These runs are respectively labeled HWC*, HWC10, and HWC20. In all cases, the step-length parameter β was initialized as 2, and was halved at iterations 800, 1200, 1400, 1500, and 1550. After iteration 1550, β was halved every 25 iterations. Detailed results are displayed in Table 3.4. As expected, HWC* revealed the smallest average POR value since it assumes knowledge of the exact optimal objective value. However, GPKC(0, 4) was competitive with HWC* despite incorporating no a prior knowledge regarding the optimal value, and significantly outperformed HWC10 and HWC20. In particular, the POR values of GPKC(0, 4) were less than or equal to those of HWC10 and HWC20 for 40 and 50 instances, respectively. Furthermore, the overall average POR value of 0.353% of HWC10 is larger than 0.33%, which is the greatest average POR value over all the modified VTVM algorithmic variants studied in Table 3.4.

For a comparison of CPU times, we define a *near-optimal objective value* as one that corresponds to a POR value of less than or equal to 1% for the MC, LP, SC, and SP problems, and less than or equal to 5% for the TR problems. (Again, note that the

Table 3.4 POR Values of HWCs and GPKC(0, 4) for Each Test Problem.

Problem	Procedure				Problem	Procedure			
	HWC*	HWC10	HWC20	GPKC (0,4)		HWC*	HWC10	HWC20	GPKC (0,4)
MC1	0	0	0	0	LP24	0.007	0.02	0.03	0.014
MC2	0	0	0	0	LP25	0.001	0.004	0.004	0.002
MC3	0	0	0	0	LP26	0.07	0.081	0.229	0.099
MC4	0	0	0	0	LP27	0.133	0.227	0.634	0.207
MC5	0	0	0	0	LP28	0.189	0.097	0.239	0.192
LP1	0.005	0.013	0.014	0.008	LP29	0.045	0.157	0.209	0.165
LP2	0.001	0.005	0.005	0.002	LP30	0.082	0.5	1.309	0.171
LP3	0	0.002	0.002	0.001	LP31	0.004	0.021	0.023	0.011
LP4	0.005	0.017	0.02	0.01	LP32	0	0.006	0.006	0.002
LP5	0.001	0.005	0.005	0.003	LP33	0	0	0	0
LP6	0.154	0.21	0.249	0.237	LP34	0.007	0.022	0.028	0.012
LP7	0.196	0.327	0.515	0.373	LP35	0.001	0.006	0.006	0.002
LP8	0.341	0.307	0.615	0.387	LP36	0.114	0.157	0.250	0.257
LP9	0.087	0.291	0.343	0.293	LP37	0.126	0.256	0.714	0.207
LP10	0.147	0.532	0.853	0.356	LP38	0.207	0.095	0.230	0.185
LP11	0	0.012	0.013	0.002	LP39	0.066	0.295	0.381	0.221
LP12	0	0.001	0.001	0	LP40	0.095	0.596	1.611	0.251
LP13	0	0	0	0	RAIL507	0.125	0.154	0.218	0.280
LP14	0.003	0.019	0.022	0.009	RAIL2586	0.137	0.093	0.137	0.277
LP15	0	0.005	0.005	0.001	RAIL4284	0.223	0.136	0.192	0.221
LP16	0.161	0.293	0.388	0.381	SPPNU16	0	0.910	1.767	0
LP17	0.219	0.405	0.699	0.355	SPPUS01	0.035	0.061	0.079	0.093
LP18	0.355	0.362	0.703	0.467	TR1	0.490	1.211	5.643	0.353
LP19	0.132	0.429	0.473	0.599	TR2	0.595	1.633	3.441	0.287
LP20	0.162	0.604	0.899	0.428	TR3	0.768	3.567	4.578	1.091
LP21	0.006	0.012	0.013	0.007	TR4	0.663	1.942	4.411	0.714
LP22	0.001	0.003	0.003	0.001	TR5	1.154	3.314	4.268	1.276
LP23	0	0.001	0.001	0.001	Average	0.133	0.353	0.663	0.191

Lagrangian dual formulation of the transportation problem is challenging to solve via subgradient approaches (see Shor, 1985).) For a meaningful comparison, we utilize the statistic RCP (relative CPU percentage) defined as: [CPU time of employed algorithm to attain a near-optimal solution] ÷ [CPU time of CPLEX 8.1 to attain the same near-optimal solution] ×100%. (The CPLEX runs utilized the dual simplex option.) Since the modified VTVM procedure yielded the best variable target value framework among the four tested alternatives (see Tables 3.2-3.3), we only exhibit the RCP values associated with the modified VTVM procedure in Table 3.5, in combination with the different direction

Table 3.5 Average RCP Values for the Modified VTVM Algorithm.

Subroutine	Problem Type					
	MC	TR	LP	SC	SP	Overall
PS	15.55	12.74	12.39	32.07	460.26	30.07
VA	10.99	11.99	11.18	26.87	682.29	36.24
ADS	16.86	30.84	9.72	27.99	688.06	38.21
GPKC(0,1)	13.65	11.39	10.42	33.41	497.69	29.78
GPKC(0,2)	14.62	11.08	10.54	33.29	472.83	29.01
GPKC(0,3)	15.58	11.04	10.64	30.27	440.86	27.84
GPKC(0,4)	15.94	11.33	10.58	31.64	490.55	29.74
GPKC(1,1)	13.79	11.23	11.38	34.59	503.31	30.74
GPKC(1,2)	16.45	10.94	10.82	31.69	465.21	29.01
GPKC(1,3)	18.95	11.04	11.21	31.02	432.92	28.31
GPKC(1,4)	18.83	11.16	11.06	32.60	430.70	28.21
GPKC(2,1)	15.22	12.23	10.80	29.30	493.09	29.88
GPKC(2,2)	16.14	10.95	10.94	31.49	475.28	29.42
GPKC(2,3)	18.47	11.03	11.19	28.79	464.98	29.30
GPKC(2,4)	18.22	11.12	11.12	29.81	442.82	28.48
GPKC(3,1)	14.09	12.23	10.66	30.67	458.23	28.48
GPKC(3,2)	14.99	11.07	10.79	31.26	452.47	28.38
GPKC(3,3)	17.03	10.91	11.13	30.79	428.53	27.89
GPKC(3,4)	16.84	11.35	11.08	30.62	433.50	28.05

and step-length strategies. Among these implemented strategies, ADS worked best for the class of LP problems. Also, VA performed best for MC and SC problems, while GPKC revealed the smallest RCP values for TR and SP. We also observed that the ADS strategy could not attain a near-optimal objective value for the instance TR5. In terms of the overall average RCP values, GPKC(0,3) yielded the best performance (27.84%) while GPKC(3,3) and GPKC(3,4) displayed the second and the third smallest RCP values (27.89% and 28.05%), respectively. However, note that the RCP values for the class of SP are very large. The reason for this is that the sizes of our set partitioning problems are too small to be advantageously solved via the Lagrangian dual formulation, as opposed to using a direct application of the commercial software CPLEX (8.1).

Also, for the sake of interest, we mention here that for the Level Algorithm used in concert with ADS (for which it exhibited the best overall performance, although it largely failed to find near-optimal solutions), the average ratios [CPU time for 2000

Table 3.6 Average POR Values for Hybrid Strategies within the Modified VTVM Procedure.

Strategy	Problem Type					Overall
	MC	TR	LP	SC	SP	
HYBRID	0	3.17	0.09	0.22	0.08	0.37
GPKC-HYBRID	0	0.95	0.13	0.24	0.05	0.20
GPKC(0,4)	0	0.74	0.15	0.26	0.05	0.19

iterations] ÷[CPU time for CPLEX 8.1 to find the same accuracy solution via dual simplex method] ×100% for each of the classes of problems MC, TR, LP, SC, and SP were respectively: 77.30%, 57.69%, 24.68%, 166.11%, and 1370.47%.

We also experimented with two hybrid strategies using VTVM in the hope that we could consolidate the relative advantages of competitive direction and step-length procedures. In the first hybrid strategy, denoted HYBRID, noting that ADS revealed the best average POR values for most classes of test problems except for the transportation problem, while GPKC(0,4) yielded the best overall performance (Table 3.3), we employed these two strategies alternately by switching whenever the target value was decreased. In a second approach, called GPKC-HYBRID, the GPKC(0,4) strategy was used for the first 500 iterations, and then HYBRID was employed for the remaining iterations. The average POR values obtained for each class of test problems (along with those for GPKC(0,4)) are displayed in Table 3.6. Both approaches produced optimal solutions for the class of MC problems. Compared with GPKC(0,4), both hybrid strategies yielded smaller POR values for the LP and SC problems, while displaying worse performances for the TR and SP problems. The GPKC-HYBRID strategy attained a better overall average POR value than did HYBRID, but it yielded a 0.01% higher POR value than that obtained by GPKC(0,4) within the modified VTVM procedure. Overall, although HYBRID appears to be a preferable strategy for general unstructured LPs, we advocate using GPKC(0,4) because of its evident robustness.

3.6 Summary and Conclusions

In this chapter, we have considered two variable target value frameworks, the Level Algorithm and VTVM, which do not require any prior information on the optimal

objective value, for solving relatively large-scale nondifferentiable optimization (NDO) problems that arise in the context of Lagrangian relaxations. Noting that the Polyak-Kelley cutting plane method (PKC) revealed a promising performance in the computations reported in Sherali et al. (2001) and Chapter 2, we first designed a generalized version GPKC, in which the cuts employ an estimate of the optimal objective value based on a lag of some $\bar{\delta}' \geq 0$ iterations and also incorporate sequential projections involving some $\bar{\delta}'' \geq 1$ previous cuts in addition to that for the current iteration. We provided convergence analyses for the Level Algorithm and VTVM when used in concert with various direction finding and step-length strategies including the pure subgradient (PS) method, the Volume Algorithm (VA), the average direction strategy (ADS), and the generalized PKC (GPKC) method. We also discussed some possible potential drawbacks for these variable target value frameworks and proposed modifications to remedy these shortcomings. As a further enhancement, whenever an improvement was attained, we performed a projected quadratic-fit line-search in which the objective values at points projected onto the feasible region are used to find a three-point-pattern.

These algorithmic compositions were extensively tested on several instances of max cut, transportation, general linear programming, set covering, and set partitioning problems. Based on our computational study, the modifications significantly improved the overall performance of each variable target value procedure. In terms of solution quality, the modified VTVM algorithm revealed the best performance when implemented with Subroutine GPKC having parameter values $\bar{\delta}' = 0$ and $\bar{\delta}'' = 4$. Also, we observed that the projected quadratic-fit line-search played a beneficial role. Moreover, the modified VTVM algorithm consistently outperformed the popular heuristic subgradient method of Held et al. (1974), even when run with consistent, relatively tight upper bounds that were slightly higher than the optimal objective value. As far as CPU times are concerned, the modified VTVM procedure in concert with the GPKC(0,3) ≡ GPKC($\bar{\delta}' = 0$, $\bar{\delta}'' = 3$) strategy revealed the best performance, providing near-optimal solutions in about 27.84% of the effort required by the commercial LP solver, CPLEX 8.1, on average. (The prescribed strategy GPKC(0,4) consumed at an average 29.74% of the effort required by CPLEX 8.1.) We also experimented with two hybrid strategies within the modified VTVM procedure, and found that while certain hybrid compositions

might be well-suited for particular types of problems, the GPKC(0,4) strategy yields the best overall performance with evident robustness.

Chapter 4:
Obviating Nondifferentiability: Perturbation Technique and Barrier-Lagrangian Dual Reformulation

4.1 Introduction

Consider an equality-constrained bounded-variables linear program in the form:

$$\text{Minimize} \quad c^T x \tag{4.1a}$$

$$\text{subject to} \quad Ax = b \tag{4.1b}$$

$$l \le x \le u\,, \tag{4.1c}$$

where A is an $m \times n$ matrix, and l, u are vectors of lower and upper bounds, respectively, on the x variables. The Lagrangian dual of (4.1) is given by

$$\text{Maximize} \quad \theta(\pi)\,, \tag{4.2a}$$

where

$$\theta(\pi) \equiv b^T\pi + \min\{c^T x - \pi^T Ax : l \le x \le u\}\,. \tag{4.2b}$$

Note that $\pi \in R^m$ is unrestricted in sign (i.e., $\Pi \equiv R^m$ in the present context), while $\theta : R^m \to R^1$ is a nondifferentiable function.

In this chapter, we develop two new approaches for solving the problem (4.2), that attempt to circumvent or obviate the nondifferentiability of θ. In Section 4.2, we utilize perturbations at nondifferentiable points so that conventional differentiable optimization methods can be directly employed as part of an algorithmic strategy for solving the problem (4.2). Next, we introduce a barrier function approach in Section 4.3 (see Bazaraa et al. (1993) for a general discussion on barrier function methods), which

provides a differentiable Lagrangian dual function that permits standard differentiable optimization techniques to be utilized. Some differentiable optimization strategies that we implement for our numerical study are briefly described in Section 4.4. Subsequently, in Section 4.5, we design two-phase algorithmic procedures in which these proposed techniques are utilized to initialize the VTVM algorithm in the hope of enhancing its numerical performance. Computational test results on the efficacy of these strategies are provided in Section 4.6, and Section 4.7 closes this chapter with a summary and conclusions.

4.2 Perturbation Technique

The basic idea behind the proposed approach is that if we can circumvent nondifferentiable points, to the extent possible, then differentiable optimization technique concepts could be employed and ascent directions would therefore be directly available. The method developed for inducing such a differentiable trajectory is called the perturbation technique (PT), and is motivated by the fact that the Lagrangian dual function θ is differentiable almost everywhere. Notice that θ is nondifferentiable only at π^k such that there exists a $j \in \{1, ..., n\}$ for which the reduced cost $c_j - (\pi^k)^T A^j = 0$, where c_j is j^{th} element of c, and A^j is j^{th} column vector of A. Although the probability that we randomly hit a nondifferentiable point is theoretically zero, this occurs routinely in practice because of the nature of the ascent process over the piecewise linear objective surface, and due to the characterization of (optimal) basic feasible solutions where the basic variables have zero reduced costs. However, as described below, we can use a perturbation technique to evade nondifferentiable points en-route toward an optimum, as possible, and contend with nondifferentiability whenever this is unavoidable, hopefully, only in the vicinity of an optimum.

Perturbation Technique

Suppose that we have a nondifferentiable point $\hat{\pi}^k = \pi^{k-1} + \lambda_{k-1}\hat{d}^{k-1}$, where π^{k-1}, λ_{k-1}, and $\hat{d}^{k-1}$ are respectively the previous iterate, step-length, and search direction.

From the condition of nondifferentiability, we have a nonempty index set $E = \{ j \in \{1, \ldots, n\} : c_j - (\hat{\pi}^k)^T A^j = 0 \}$. To find a suitable differentiable point π^k in the neighborhood of $\hat{\pi}^k$, we want to perturb $\hat{\pi}^k$ to $\pi^k = \hat{\pi}^k + \varepsilon\tilde{\pi}$ such that $c_j - (\pi^k)^T A^j = c_j - (\hat{\pi}^k)^T A^j - \varepsilon\tilde{\pi}^T A^j \neq 0$ for $j = 1, ..., n$. We can pick $\varepsilon > 0$ small enough for this to occur so long as we have

$$\tilde{\pi}^T A^j \neq 0, \quad \forall \ j \in E, \tag{4.3a}$$

and

$$0 < \varepsilon < \bar{\varepsilon} \equiv \min\{\varepsilon' > 0 : c_j - (\hat{\pi}^k + \varepsilon'\tilde{\pi})^T A^j = 0 \text{ for some } j \in E^C\}, \tag{4.3b}$$

where E^C denotes the complement of E with respect to $\{1, \ldots, n\}$.

Once we have $\tilde{\pi}$ and ε satisfying the conditions (4.3a) and (4.3b), we uniquely obtain

$$x^k = \arg\min\{(c^T - (\hat{\pi}^k + \varepsilon\tilde{\pi})^T A)x : x \in X\}.$$

Notice that $\theta_k - \theta(\hat{\pi}^k) = (g^k)^T(\pi^k - \hat{\pi}^k) = \varepsilon(g^k)^T\tilde{\pi}$. Hence, we will obtain an improvement $\theta_k > \theta_{k-1}$ in the objective function whenever $\theta(\hat{\pi}^k) - \theta_{k-1} > \theta(\hat{\pi}^k) - \theta_k = -\varepsilon(g^k)^T\tilde{\pi}$. Thus, in case $(g^k)^T\tilde{\pi} < 0$, and given that $\theta(\hat{\pi}^k) > \theta_{k-1}$, we could further restrict ε to satisfy

$$\varepsilon < \frac{\theta_{k-1} - \theta(\hat{\pi}^k)}{(g^k)^T\tilde{\pi}}.$$

Consequently, given $\tilde{\pi}$ satisfying the condition (4.3a), we select ε such that

$$\varepsilon = \begin{cases} \eta \min\{\bar{\varepsilon}, [\theta_{k-1} - \theta(\hat{\pi}^k)] / (g^k)^T \tilde{\pi}\} & \text{if } (g^k)^T \tilde{\pi} < 0 \\ & \quad \text{and given } \theta(\hat{\pi}^k) > \theta_{k-1}, \\ \eta \bar{\varepsilon} & \text{otherwise,} \end{cases} \tag{4.4a}$$

(4.4b)

where $0 < \eta < 1$.

Toward designing such a perturbation scheme, we could commence with $\tilde{\pi} \equiv \hat{\pi}^k$. If the condition (4.3a) holds for $\tilde{\pi} = \hat{\pi}^k$, we can pick ε according to (4.4), and then we would have a new differentiable point of the form $\pi^k = \hat{\pi}^k (1 + \varepsilon)$, a simple extension of $\hat{\pi}^k$ itself. Alternatively, we could select $\tilde{\pi} = \hat{d}^{k-1}$, the previous search direction (or $\tilde{\pi} = -\hat{d}^{k-1}$), so that we would effectively test if slightly overstepping (or understepping) in the previous iteration would yield a nondifferentiable point. Otherwise, a revised perturbation vector $\tilde{\pi}_{new}$ that comes closer to satisfying the condition (4.3a) can be obtained as follows, yielding a finite iterative scheme for achieving (4.3a).

Given a current $\tilde{\pi}$, consider a perturbation of the form $\tilde{\pi}_{new} = \tilde{\pi} + \varepsilon_0 A^r$ for an arbitrary $r \in E_0 \equiv \{ j \in E : \tilde{\pi}^T A^j = 0 \}$, where $\varepsilon_0 > 0$. Then, we have

$$\tilde{\pi}_{new}^T A^j = \tilde{\pi}^T A^j + \varepsilon_0 (A^r)^T A^j, \tag{4.5}$$

where $\tilde{\pi}^T A^j \neq 0 \ \forall \ j \in E - E_0$, and

$$\tilde{\pi}_{new}^T A^r = \varepsilon_0 \left\| A^r \right\|^2 > 0. \tag{4.6}$$

From (4.5) and (4.6), if

$$\tilde{\pi}_{new}^T A^j = \tilde{\pi}^T A^j + \varepsilon_0 (A^r)^T A^j \neq 0, \forall \ j \in E - E_0, \tag{4.7}$$

then we can obtain a new index set $E_0^{new} \equiv \{ j \in E : \tilde{\pi}_{new}^T A^j = 0 \}$, which is a proper subset of E_0. Notice that (4.7) can be zero for some $j \in E - E_0$ only when $\tilde{\pi}^T A^j$ and $(A^r)^T A^j$ are of opposite sign. Hence, let $Z \equiv \{ j \in E - E_0 : (\tilde{\pi}^T A^j)((A^r)^T A^j) < 0 \}$. If $Z = \varnothing$, $\varepsilon_0 > 0$ can be arbitrarily chosen (e.g., we can take $\varepsilon_0 = 1$). Else, in order to make the condition (4.7) hold true, we select

$$\varepsilon_0 = \eta_0 \min \{ -\tilde{\pi}^T A^j / (A^r)^T A^j : j \in Z \}, \tag{4.8}$$

where $0 < \eta_0 < 1$. Reiterating this process until we have $E_0^{new} = \varnothing$, we would obtain a $\tilde{\pi}$ satisfying the condition (4.3a). The resulting perturbation routine is summarized below.

Perturbation Routine

Step 1: Given a nondifferentiable point $\hat{\pi}^k$, define the index set $E \equiv \{ j \in \{1, \ldots, n\} : c_j - (\hat{\pi}^k)^T A^j = 0 \}$. If $E = \varnothing$, return $\pi^k = \hat{\pi}^k$. Otherwise, let $\tilde{\pi} \equiv \hat{\pi}^k$ and $E_0 \equiv \{ j \in E : \tilde{\pi}^T A^j = 0 \}$, and proceed to Step 2.

Step 2: If $E_0 = \varnothing$, return $\pi^k = \hat{\pi}^k + \varepsilon \tilde{\pi}$, where $\varepsilon > 0$ is determined by (4.4). Otherwise, proceed to Step 3.

Step 3: Pick any $r \in E_0$. Let $Z \equiv \{ j \in E - E_0 : (\tilde{\pi}^T A^j)((A^r)^T A^j) < 0 \}$. If $Z = \varnothing$, put $\varepsilon_0 = 1$. Otherwise, select ε_0 according to (4.8). Let $\tilde{\pi} \leftarrow \tilde{\pi} + \varepsilon_0 A^r$ and $E_0 \leftarrow E_0 - \{ j \in E_0 : \tilde{\pi}^T A^j \neq 0 \}$. Return to Step 2.

Remark 4.1: As mentioned earlier, instead of letting $\tilde{\pi} \equiv \hat{\pi}^k$ in Step 1, we can initialize $\tilde{\pi} \equiv \pm \hat{d}^{k-1}$ to test if overstepping or understepping in the previous iteration would yield a nondifferentiable point. For the selection of $r \in E_0$ in Step 3, we initially prioritize A^j, $j = 1, \ldots, n$, from the largest number of nonzero elements to the least. Then, $r \in E_0$ is chosen according to this priority in the hope that A^r will yield $(A^r)^T A^j \neq 0$ for many $j \in E_0$, thereby potentially reducing $|E_0|$ faster (see Step 3 in the Perturbation Routine).

Now, whenever we encounter a nondifferentiable update $\hat{\pi}^k$, i.e., a nonempty index set E, we can obtain a new iterate π^k via this perturbation routine at which a solution of the Lagrangian subproblem in (4) is uniquely determined, and hence, the corresponding subdifferential is a singleton. Note that this subgradient is indeed a steepest ascent direction at π^k . Therefore, we can utilize conventional differentiable optimization strategies to obtain the next update $\hat{\pi}^{k+1}$, as further exemplified in Section 4 below.

4.3 Barrier-Lagrangian Dual Reformulation

In this section, we propose a two-step Lagrangian dual reformulation, called the *Barrier-Lagrangian Dual Reformulation* (BLR), via a barrier function. For $\mu > 0$ and sufficiently small, consider the following barrier problem, based on Frisch's logarithmic barrier function applied to the bounding constraints in (4.1c) (see Bazaraa et al. (1993), for example).

BP: Minimize $c^T x - \mu \sum_{j=1}^{n} [\ln(x_j - l_j) + \ln(u_j - x_j)]$ (4.9a)

subject to $Ax = b$. (4.9b)

We know from the barrier function theory that as $\mu \to 0^+$, the trajectory of optimal solutions to (4.9) approaches an optimum for LP.

Now, let us consider solving BP via a Lagrangian dual approach for a fixed, relatively small value of $\mu > 0$, given our intent to derive a near-optimal solution to LP. This yields the dual problem LDBP given below.

LDBP: Maximize $\bar{\theta}(\bar{\pi})$, (4.10a)

where

$$\bar{\theta}(\bar{\pi}) = b^T \bar{\pi} + \min_x \left\{ (c^T - \bar{\pi}^T A)x - \mu \sum_{j=1}^{n} [\ln(x_j - l_j) + \ln(u_j - x_j)] \right\}. \qquad (4.10b)$$

Since the objective function of the Lagrangian subproblem in (4.10b) is strictly convex, it has a unique optimum, given that a critical point exists as verified below. Differentiating and setting equal to zero to compute this critical point, we obtain

$$\bar{c}_j - \mu \left[\frac{1}{x_j - l_j} - \frac{1}{u_j - x_j} \right] = 0, \quad \forall j,$$

where $\bar{c}_j \equiv c_j - \bar{\pi}^T A^j$, $\forall j$, and where A^j is the jth column vector of A. Simplifying, we have,

$$\bar{c}_j x_j^2 - x_j[\bar{c}_j(l_j + u_j) + 2\mu] + [\bar{c}_j l_j u_j + \mu(l_j + u_j)] = 0, \ \forall j. \qquad (4.11)$$

If $\bar{c}_j = 0$, then (4.11) yields

$$x_j = \frac{l_j + u_j}{2} \equiv \bar{x}_j, \text{ say.}$$

Otherwise, solving the quadratic equation (4.11), we get

$$x_j = \bar{x}_j \equiv \frac{\left[\bar{c}_j(l_j + u_j) + 2\mu\right] \pm \sqrt{\bar{c}_j^2 (u_j - l_j)^2 + 4\mu^2}}{2\bar{c}_j}. \qquad (4.12)$$

Note that from (4.12), we obtain

$$\bar{x}_j - l_j \equiv \frac{\left[\bar{c}_j(u_j - l_j) + 2\mu\right] \pm \sqrt{\bar{c}_j^2 (u_j - l_j)^2 + 4\mu^2}}{2\bar{c}_j} \qquad (4.13a)$$

and

$$u_j - \bar{x}_j \equiv \frac{\left[\bar{c}_j(u_j - l_j) - 2\mu\right] \mp \sqrt{\bar{c}_j^2(u_j - l_j)^2 + 4\mu^2}}{2\bar{c}_j}. \tag{4.13b}$$

Because we must have $l_j < \bar{x}_j < u_j$, $\forall j$, at optimality to (4.10b), observe from (4.13) that if $\bar{c}_j > 0$, then either $\pm$ is valid for (4.13a) but we must use the plus-case in (4.13b), which corresponds to the minus-case in (4.12). Likewise, if $\bar{c}_j < 0$, then we must use the minus-case in (4.13a), while either case is legitimate for (4.13b), again implying the minus-case for (4.12). This leads to the following solution $x = \bar{x}$ for (4.10b).

$$\bar{x}_j = \frac{\left[\bar{c}_j(l_j + u_j) + 2\mu\right] - \sqrt{\bar{c}_j^2(u_j - l_j)^2 + 4\mu^2}}{2\bar{c}_j}, \qquad \text{if } \bar{c}_j \neq 0, \tag{4.14a}$$

$$\bar{x}_j = \frac{l_j + u_j}{2}, \qquad \text{if } \bar{c}_j = 0. \tag{4.14b}$$

Observe that $\bar{x}_j$ is a continuous function of $\bar{c}_j$ in that, using L'Hospital's Rule, the limiting value of (4.14a) as $\bar{c}_j \to 0$ equals the value in (4.14b). Moreover, by the uniqueness of the solution in (4.10b), $\bar{\theta}$ is a differentiable function of $\bar{\pi}$, with

$$\nabla\bar{\theta}(\bar{\pi}) = b - A\bar{x}, \tag{4.15}$$

where $\bar{x}$ is given by (4.14), with $\bar{c}_j = c_j - \bar{\pi}^T A^j$, $\forall j$. By viewing $\bar{x}$ as a function of $\bar{\pi}$ via (4.14), we can also verify that $\bar{\theta}$ is twice-differentiable, since by applying the definition of differentiation and using L'Hospital's Rule, we can verify that (see Appendix I),

$$\left.\frac{d\bar{x}_j(\bar{c}_j)}{d\bar{c}_j}\right|_{\bar{c}_j=0} = \lim_{\bar{c}_j \to 0} \frac{d\bar{x}_j}{d\bar{c}_j} = \frac{-(u_j - l_j)^2}{8\mu}.$$

Nonetheless, in keeping with our motivation to maintain simplicity in implementation, we experiment with several conjugate gradient strategies to solve LDBP for a given $\mu > 0$, instead of using Newton or quasi-Newton methods that would require large-scale linear systems to be solved at each iteration.

4.4 Evaluated Differentiable Optimization Strategies

In this section, we consider several conjugate gradient methods for solving LD via the perturbation technique and for solving LDBP. For both problems, let π^k, g^k, and d^k denote the iterate, (sub)gradient, and search direction, respectively, at iteration k. Then, commencing with some iterate π^1, we adopt the initial search direction $d^1 = g^1$. At any iteration $k \geq 2$, we derive a search direction via the conjugate gradient strategy

$$d^k = g^k + \alpha_k d^{k-1},$$

where α_k is a suitable deflection multiplier. The next iterate is then computed as

$$\bar{\pi}^{k+1} = \bar{\pi}^k + \lambda_k \frac{d^k}{\|d^k\|},$$

where a step-size $\lambda_k > 0$ is obtained via a line-search along the direction d^k. (We employ an inexact line-search here via a single quadratic-fit.) Four choices of the deflection multiplier α_k that are explored in Sherali and Ulular (1990), along with Sherali and Ulular's (1989) average direction strategy (ADS), are tested in our experiments in the next section. Specifically, based on a quadratic approximation of the objective function, Hestenes and Stiefel (1952) derived

$$\alpha_k^{HS} = \frac{(g)^T q^k}{(d^{k-1})^T q^k},$$

where $p^k \equiv \pi^k - \pi^{k-1}$ and $q^k \equiv g^k - g^{k-1}$. Assuming exact line-searches, this reduces to Polak and Ribiere's (1969) deflection parameter:

$$\alpha_k^{PR} = \frac{(g^k)^T q^k}{\left\| g^{k-1} \right\|^2}.$$

Furthermore, under an assumption of quadraticity, this simplifies to Fletcher and Reeves' (1964) formula:

$$\alpha_k^{FR} = \frac{\left\| g^k \right\|^2}{\left\| g^{k-1} \right\|^2}.$$

Based on a scaled quasi-Newton method, Sherali and Ulular (1990) alternatively prescribed the deflection parameter

$$\alpha_k^{SU} = -\frac{(g^k)^T (q^k + d^{k-1})}{(q^k)^T d^{k-1}}.$$

Finally, Sherali and Ulular (1989) proposed the ADS strategy, given by

$$\alpha_k^{ADS} = \frac{\left\| g^k \right\|}{\left\| d^{k-1} \right\|},$$

and demonstrated a promising performance for solving the nondifferentiable Lagrangian dual of LP itself. In addition to these five conjugate gradient strategies, we consider the

memoryless BFGS method of Shanno (1978) in our computational experiments, for which a search direction is prescribed as

$$d^k = -\frac{(p^k)^T q^k}{(q^k)^T q^k} g^k - \left(2\frac{(p^k)^T g^k}{(p^k)^T q^k} - \frac{(q^k)^T g^k}{(q^k)^T q^k} \right) p^k + \frac{(p^k)^T g^k}{(q^k)^T q^k} q^k .$$

Note that, for the perturbation technique, if $q^k = 0$ (or practically $\left\| q^k \right\| < \varepsilon$ for some tolerance $\varepsilon > 0$), we use the (sub)gradient as the next search direction, i.e., $d^k = g^k$. We employ a single quadratic-fit line-search along with the restarting criteria of Powell (1977) for the foregoing conjugate (sub)gradient direction and memoryless quasi-Newton strategies.

4.5 Algorithmic Procedures

The overall algorithmic procedure consists of two sequential phases. In the first phase, the foregoing schemes that circumvent or obviate nondifferentiability of θ are utilized in the hope that a relatively good solution is found. Then, in the second phase of the procedure, we switch to a theoretically convergent NDO algorithm, which is a composition of a modified variable target value method (VTVM) of Sherali et al. (2000) and a generalized Polyak-Kelley cutting plane strategy (GPKC) as proposed in Chapter 3. Recall that this combination (VTVM-GPKC) revealed the most promising performance when extensively tested on a variety of problems in Section 3.5.

At the end of the first phase, after some K iterations, say, we estimate the optimal objective value for LD using a quadratic approximation for the original Lagrangian dual function θ in (4.2). This estimate is used as the initial target value w_1 for the second phase. Suppose that $\hat{\theta}$ is a quadratic approximation for θ and given by

$$\hat{\theta}(\pi) = h_0 + h^T \pi + \frac{1}{2} \pi^T H \pi , \qquad (4.16)$$

where H is a negative definite diagonal matrix. Maximizing $\hat{\theta}$ gives an optimal solution and the optimal objective value as

$$\hat{\pi} = -H^{-1}h \text{ and } \hat{w} = \hat{\theta}(\hat{\pi}) = h_0 - \frac{1}{2}h^T H^{-1}h, \tag{4.17}$$

respectively. Now, suppose that we evaluate $\theta(\pi^i)$ and $g^i \in \partial\theta(\pi^i)$ for $i = 1$ and K, where $\partial\theta(\pi)$ is the subdifferential of θ at π. Using the approximation (4.16), we equate

$$g^K - g^1 = \nabla\hat{\theta}(\pi^K) - \nabla\hat{\theta}(\pi^1) = H(\pi^K - \pi^1), \tag{4.18}$$

in order to estimate the assumed diagonal Hessian H. To maintain strict concavity of $\hat{\theta}$, if any diagonal element of H is nonnegative or indeterminable from (4.18), we take it as -1. This gives us the desired estimate for the diagonal negative definite matrix H. Furthermore, we estimate h_0 and h by equating θ_K and g^K to $\hat{\theta}(\pi^K)$ and $\nabla\hat{\theta}(\pi^K)$, respectively, according to

$$\hat{\theta}(\pi^K) = h_0 + h^T\pi^K + \frac{1}{2}(\pi^K)^T H\pi^K = \theta_K \text{ and } \nabla\hat{\theta}(\pi^K) = h + H\pi^K = g^K. \tag{4.19}$$

Substituting h_0, h, and H as obtained from (4.18) and (4.19) into (4.17), we compute an initial target value $w_1 \equiv \hat{w}$ for the VTVM-GPKC phase. Note from (4.19) that since $\nabla\hat{\theta}(\pi^K) = g^K \neq 0$ while $\nabla\hat{\theta}(\hat{\pi}) = 0$ by definition, we have that $\hat{w} = \hat{\theta}(\hat{\pi}) > \hat{\theta}(\pi^K) = \theta_K$, i.e., $\hat{w} > \theta_K$ as desired.

We also experimented with using an initial target value as given by $w_1 \equiv \theta_K + p(\theta_K - \theta_1)$, where $p > 0$, based on a simple further projected estimated improvement of $p \times 100\%$ in the objective value.

Let us denote these two foregoing initial target value determination methods by QUAD and PROJ, respectively. Furthermore, assuming that the first phase provides a

relatively good solution, we limit the number of iterations for the second phase to 1000.

For the sake of completeness, we describe the overall two-phase algorithmic procedure below, including the options of the perturbation technique (PT) or the barrier-Lagrangian dual reformulation (BLR) in Phase I, and the VTVM-GPKC method recommended in Chapter 3 that is used in Phase II.

Phase I (PT)

Step 1: Select an initial iterate π^1 at which θ is differentiable (use the perturbation routine in Section 2 if necessary). Pick a termination tolerance $\varepsilon_{tol} = 10^{-6}$ for the norm of the (sub)gradient, and a maximum number of iterations $K = 100$. Compute x^1 and g^1. Set the iteration counter $k = 1$ and the restart indicator RESET = 0.

Step 2: If $\left\| g^k \right\| \le \varepsilon_{tol}$ or $k = K$, then stop the PT Phase, and go to Phase II with the initial solution $\pi^1 \equiv \pi^k$ and the initial target value w_1 as obtained above via the QUAD or PROJ methods.

Step 3: If RESET = 0, put $d^k = g^k$. Otherwise, compute a search direction based on the employed (deflected (sub)gradient) direction finding strategy.

Step 4: If $k \ge 2$, RESET ≥ 1, and any of the following two restarting criteria is violated (see Bazaraa et al. 1993):

i) $\left| (g^k)^T g^{k-1} \right| \le 0.2 \left\| g^k \right\|^2$

ii) $0.8 \left\| g^k \right\|^2 \le (d^k)^T g^k \le 1.2 \left\| g^k \right\|^2$,

then put RESET = 0 and return to Step 3. Else, perform a single quadratic-fit line-search along d^k to find a new iterate π^{k+1} (use the perturbation routine as necessary), and increment $k \leftarrow k+1$. Increment RESET $\leftarrow$ RESET+1, and if RESET= n, put RESET=0.

Step 5: Compute x^k and g^k, and return to Step 2.

Phase I (BLR Method)

Step 1: Select an initial iterate π^1, a termination tolerance $\varepsilon_{tol} = 10^{-6}$ for the norm of the gradient, a barrier parameter $\mu > 0$, and a maximum number of iterations $K = 100$ for the BRL Phase. Compute $\bar{x}^1$ and $\nabla\bar{\theta}(\pi^1)$ according to (15) and (16), respectively. Set the iteration counter $k = 1$ and the restart indicator RESET = 0.

Step 2: If $\left\|\nabla\bar{\theta}(\pi^k)\right\| \le \varepsilon_{tol}$ or $k = K$, then stop the BRL Phase, and go to Phase II with the initial solution $\pi^1 \equiv \pi^k$ and the initial target value w_1 as obtained above via the QUAD or PROJ methods.

Step 3: If RESET = 0, put $d^k = \nabla\bar{\theta}(\pi^k)$. Otherwise, compute a search direction based on the employed (deflected gradient) direction finding strategy.

Step 4: If $k \ge 2$, RESET ≥ 1, and any of the following two restarting criteria is violated (see Bazaraa et al. 1993):

i) $\left|(\nabla\bar{\theta}(\pi^k))^T \nabla\bar{\theta}(\pi^{k-1})\right| \le 0.2\left\|\nabla\bar{\theta}(\pi^k)\right\|^2$

ii) $0.8\left\|\nabla\bar{\theta}(\pi^k)\right\|^2 \le (d^k)^T \nabla\bar{\theta}(\pi^k) \le 1.2\left\|\nabla\bar{\theta}(\pi^k)\right\|^2$,

then put RESET = 0 and return to Step 3. Else, perform a single quadratic-fit line-search along d^k to find a new iterate π^{k+1}, and increment $k \leftarrow k+1$. Increment RESET $\leftarrow$ RESET+1, and if RESET= n, put RESET=0.

Step 5: Compute $\bar{x}^k$ and $\nabla\bar{\theta}(\pi^k)$ via (4.14) and (4.15), and return to Step 2.

Phase II (VTVM-GPKC Method)

Initialization Step: Select $\varepsilon_{tol} = 10^{-6}$, $\tilde{\varepsilon} > 0$, $k_{\max}$, $\beta \in (0,1]$, $\sigma \in (0,1/3]$, r, $\bar{r}$, $\bar{\tau}$, $\bar{\gamma}$, and $\eta \in (0,1]$. (Recommended values for these parameters are $\tilde{\varepsilon} = 0.1$, $k_{\max} = 1000$, $\beta = 0.8$, $\sigma = 0.15$, $r = 0.1$, $\bar{r} = 1.1$, $\bar{\tau} = 75$, $\bar{\gamma} = 20$, and $\eta = 0.75$.) Determine $\theta_1 \equiv \theta(\pi^1)$ and g^1. Set the incumbent vectors $\bar{\pi} = \pi^1$, $\bar{g} = g^1$, and the incumbent objective value $z_1 = \theta_1$. If $\left\|g^1\right\| \le \varepsilon_{tol}$, terminate the algorithm. Initialize $l = 1$, $k = 1$, $\tau = 0$, $\gamma = 0$, $\Delta = 0$, and the reset indicator RESET=1. Set

$\varepsilon_1 = \sigma(w_1 - \theta_1)$.

Step 1: Call Subroutine GPKC (described below) to obtain the next iterate π^{k+1}. Increment $\tau \leftarrow \tau + 1$ and $k \leftarrow k+1$. Compute $\theta_k \equiv \theta(\pi^k)$ and g^k. Put RESET=0.

Step 2: If $\theta_k > z_{k-1}$, set $\Delta \leftarrow \Delta + (\theta_k - z_{k-1})$, $z_k = \theta_k$, $\overline{\pi} = \pi^k$, $\overline{g} = g^k$, $\gamma = 0$, and proceed to Step 3. Otherwise, put $z_k = z_{k-1}$, increment $\gamma \leftarrow \gamma + 1$, and go to Step 4.

Step 3: If $k > k_{\max}$ or $\left\| g^k \right\| \le \varepsilon_{tol}$, terminate the algorithm. If $z_k \ge w_l - \varepsilon_l$, go to Step 5. If $\tau \ge \overline{\tau}$, go to Step 6. Else, return to Step 1.

Step 4: If $k > k_{\max}$, terminate the algorithm. If $\gamma \ge \overline{\gamma}$ or $\tau \ge \overline{\tau}$, go to Step 6. Else, return to Step 1.

Step 5: Compute $w_{l+1} = z_k + \max\{\varepsilon_l + \eta\Delta, r|z_k|\}$, and $\varepsilon_{l+1} = \max\{(w_{l+1} - z_k)\sigma, \tilde{\varepsilon}\}$. If $k \le 500$, update $\eta \leftarrow 2\eta$; otherwise, set $\eta = 0.75$. If $r|z_k| > \varepsilon_l + \eta\Delta$, update $r \leftarrow r / \overline{r}$. Put $\tau = 0$, $\Delta = 0$, and increment $l \leftarrow l+1$. Return to Step 1.

Step 6: Compute $w_{l+1} = \dfrac{(z_k + \varepsilon_l) + w_l}{2}$, and $\varepsilon_{l+1} = \max\{(w_{l+1} - z_k)\sigma, \tilde{\varepsilon}\}$. If $k \le 500$, update $\eta \leftarrow \max\{\eta/2, 0.75\}$; otherwise, set $\eta = 0.75$. If $\gamma \ge \overline{\gamma}$, then set $\overline{\gamma} \leftarrow \min\{\overline{\gamma} + 10, 50\}$. If $(w_l - w_{l+1}) \le 0.1$, then set $\beta \leftarrow \max\{\beta/2,\ 10^{-6}\}$. Put $\gamma = 0$, $\tau = 0$, $\Delta = 0$, and $l \leftarrow l+1$. Reset $\pi^k = \overline{\pi}$, $\theta_k = z_k$, $g^k = \overline{g}$, and put RESET=1. Return to Step 1.

Subroutine GPKC

Step i: Let $\delta = k - \overline{k}$, where $\overline{k}$ is the most recent iteration at which the variable target value algorithm was reset (i.e. RESET=1), and accordingly, let $\delta'' = \min\{\delta, 4\}$. Compute $\hat{\theta}^* = \theta_k + \beta(w_l - \theta_k)$.

Step ii: Set $\psi_1 = \dfrac{\hat{\theta}^* - \theta_k}{\left\| g^k \right\|^2}$, $\psi_2 = 0$, and $\pi^{k+1} = \pi^k + \psi_1 g^k$. If RESET=1, put $\overline{k} = k$ and exit the subroutine.

Step iii: Compute $\xi_1 = \hat{\theta}^* - \theta_k + (\pi^k)^T g^k$ and $\xi_2 = \hat{\theta}^* - \theta_{k-1} + (\pi^{k-1})^T g^{k-1}$. If $\xi_2 \leq (\pi^{k+1})^T g^{k-1}$, go to Step vi.

Step iv: Put $\psi_2 = \dfrac{\xi_2 - (\pi^k)^T g^{k-1}}{\left\| g^{k-1} \right\|^2}$, and let $\pi^{k+1} = \pi^k + \psi_2 \left(g^{k-1} \right)^T$. If $\psi_2 > 0$ and $(\hat{\pi}^{k+1})^T g^k \geq \xi_1$, go to Step vi.

Step v: Compute $\psi_0 = \left\| g^k \right\|^2 \left\| g^{k-1} \right\|^2 - \left((g^{k-1})^T g^k \right)^2$. If $\psi_0 < 10^{-6}$, put $\pi^{k+1} = \pi^k + \psi_1 g^k$ and exit the subroutine. Otherwise, compute

$$\psi_1 = \left[\left\| g^{k-1} \right\|^2 \left(\xi_1 - (\pi^k)^T g^k \right) - \left((g^{k-1})^T g^k \right) \left(\xi_2 - (\pi^k)^T g^{k-1} \right) \right] \Big/ \psi_0 \text{, and}$$

$$\psi_2 = \left[\left\| g^k \right\|^2 \left(\xi_2 - (\pi^k)^T g^{k-1} \right) - \left((g^{k-1})^T g^k \right) \left(\xi_1 - (\pi^k)^T g^k \right) \right] \Big/ \psi_0 .$$

Put $\pi^{k+1} = \pi^k + \psi_1 g^k + \psi_2 g^{k-1}$ and proceed to Step vi.

Step vi: If $\delta'' = 1$, exit the subroutine. Otherwise, put $i = 2$ and proceed to Step vii.

Step vii: Compute $\xi_3 = \hat{\theta}^* - \theta_{k-i} + (\pi^{k-i})^T g^{k-i}$ and let $\psi_3 = \xi_3 - (\pi^{k+1})^T g^{k-i}$. If $\psi_3 \leq 0$, go to Step ix. Otherwise, proceed to Step viii.

Step viii: Put $\bar{\pi}^{k,i} = \pi^{k+1} + \dfrac{\psi_3}{\left\| g^{k-i} \right\|^2} g^{k-i}$. If $(\bar{\pi}^{k,i})^T g^k \geq \xi_1$ and $(\bar{\pi}^{k,i})^T g^{k-1} \geq \xi_2$, put $\pi^{k+1} = \bar{\pi}^{k,i}$.

Step ix: If $i = \delta''$, exit the subroutine. Otherwise, increment $i \leftarrow i + 1$ and return to Step vii.

Remark 4.2: Often, in the context of solving LP, a (near-optimal) primal solution is also desirable besides obtaining an optimal dual solution along with the optimal objective value. Commencing with $\bar{x}^K$ as obtained after performing some K iterations using the BLR option in Phase I, we could employ an ergodic primal recovery strategy while executing the VTVM-GPKC algorithm in Phase II in order to derive such a primal solution for LP. (See Shor (1985), Sherali and Choi (1996), and Larsson et al. (1999) for several ergodic primal recovery strategies.) In case a more accurate primal solution is desired (as for example when generating cuts for discrete problems based on a given LP

relaxation), the resulting primal solution obtained via the foregoing process can be used as a warm-start for an interior point or simplex-based algorithm (after applying a purification scheme to obtain a vertex solution in the latter case as described in Bazaraa et al. (1990), for example). In the next section, we provide some computational results pertaining to such a primal recovery process as well, in combination with the proposed two-phase dual optimization scheme that employs the BLR option in Phase I.

4.6 Computational Study

The proposed algorithmic procedures were implemented for solving two types of test problems: transportation and general linear programming problems. Twenty transportation problems having various sizes and extent of degeneracy, TR1-TR20, were randomly generated as in Chapter 2. However, notice that we did not exploit the structure of the constraints in the transportation problem when formulating the corresponding Lagrangian dual problem, i.e., we simply dualized both the demand and supply balance constraints in order to test the proposed procedures. Also, similar to generating the inequality constrained problems as described in Chapter 2, twenty general linear programming problems, LP1-LP20, formulated as LP in (4.1) using $l = 0$ and $u = e$, were generated, having a variety of sizes and degrees of degeneracy as follows.

For the primal solution, some m components of x were randomly generated on [0, 1], with some $\rho_P\%$ of these being set at 0 or 1, while the rest of the x variables were necessarily randomly set to 0 or 1. The coefficient matrix A was generated based on a uniform distribution on $[-5, 5]$, having a density of 0.05 and at least one nonzero component in each row. The right-hand side vector b was chosen as $b = Ax$. For the dual solution π associated with equality constraints, the components π_i, were uniformly generated on $[-10, 10]$. The dual variables γ and μ corresponding to the lower and upper bounding constraints, respectively, were generated as follows. Corresponding to the x_j variables that were necessarily set at one above, we put $\mu_j = 0$ and generated γ_j uniformly on $[0, 5]$, randomly setting some $\rho_D\%$ of these at zeros. Likewise, corresponding to the x_j variables that were necessarily set at one above, we put $\gamma_j = 0$

Table 4.1 Summary of Test Problems for the Computational Study of Two-Phase Procedures.

Problem Type	Prob	Row/Column	CPLEX solution		Problem Type	Prob	Row/Column	CPLEX solution	
			Optimal Value	CPU time (sec)				Optimal value	CPU time (sec)
Transportation (5%, 5%)	TR1	800/160000	2215.08	128.56	Linear Program (5%, 5%)	LP1	500/1000	-1703.83	44.89
	TR2	1000/250000	1499.65	272.27		LP2	1000/3000	-4724.01	1226.86
	TR3	1200/360000	-895.093	567.47		LP3	1000/5000	-1222.18	1479.31
	TR4	1400/490000	305.108	881.82		LP4	2000/3000	3078.64	9466.68
	TR5	1600/640000	-204.755	1335.25		LP5	2000/5000	3105.01	12986.7
Transportation (5%, 25%)	TR6	800/160000	681.531	125.04	Linear Program (5%, 25%)	LP6	500/1000	-623.999	48.75
	TR7	1000/250000	-571.108	221.71		LP7	1000/3000	-2748.89	1154.44
	TR8	1200/360000	-1779.95	547.84		LP8	1000/5000	-5306.73	1708.88
	TR9	1400/490000	-1278.88	627.79		LP9	2000/3000	-4899.56	10375.8
	TR10	1600/640000	1614.92	1126.82		LP10	2000/5000	-2492.58	23079.6
Transportation (25%, 5%)	TR11	800/160000	-601.262	79.58	Linear Program (25%, 5%)	LP11	500/1000	-1672.58	43.78
	TR12	1000/250000	43.413	168.35		LP12	1000/3000	-271.81	966.13
	TR13	1200/360000	-2308.51	349.38		LP13	1000/5000	1375.16	1343.07
	TR14	1400/490000	1440.23	573.34		LP14	2000/3000	-2107.96	9015.16
	TR15	1600/640000	-1163.07	986.80		LP15	2000/5000	-3456.14	14872.2
Transportation (25%, 25%)	TR16	800/160000	553.837	85.84	Linear Program (25%, 25%)	LP16	500/1000	-338.54	45.62
	TR17	1000/250000	-1361.69	221.17		LP17	1000/3000	1099.29	1173.57
	TR18	1200/360000	354.879	296.13		LP18	1000/5000	-11369.5	1553.11
	TR19	1400/490000	-2139	402.69		LP19	2000/3000	1121.03	6378.62
	TR20	1600/640000	2144.66	600.79		LP20	2000/5000	-5871.56	43708.8

and generated μ_j on uniformly on $[0, 5]$, randomly setting some $\rho_D\%$ of these at zero. The remaining γ and μ variable components were set at zero. We computed the coefficient vector of the objective function, c, as $c = \pi A + \gamma - \mu$.

Particular specifications for these test problems are summarized in Table 4.1, where the two percentage values given in parentheses respectively indicate the degrees of primal and dual degeneracy induced at optimality. We also display the CPU times consumed by the commercial linear programming solver, CPLEX Version 8.1, to solve this test bed of problems. Again, all runs were made on a PentiumIII/667Mhz computer and the origin of the corresponding dual space was used as the initial solution for each algorithm and problem combination. Based on our preliminary test, we used $p = 0.6$ for the target value initialization method PROJ. Furthermore, in order to compare the quality of solutions produced by the different methods, we report a percentage optimality ratio (POR) value (see Section 2.5 for the definition of POR).

Table 4.2 presents the POR values obtained after executing Phase I for the PT and BLR procedures in conjunction with the six deflection strategies described in Section 4.4, using an iteration limit of $K = 100$. For both PT and BLR, the rank-order of the overall performance was SU, FR, ADS, PR, BFGS, and HS from the best to the worst. Note that the PT and BLR techniques respectively yielded overall average POR values of 0.74% and 0.73% when implemented in concert with the SU strategy. Moreover, the BLR technique revealed the best respective POR values of 1.17% and 0.29% when used in conjunction with SU for both the classes of LP and TR problems. Also, the strategy SU displayed the best POR value of 1.15% for LP problems when implemented with PT. However, with the PT procedure, the ADS strategy yielded the least POR value of 0.29% for TR problems, while SU was the second best, producing a 0.03% greater POR value than ADS. Considering the entire test cases together, the BLR technique slightly, but consistently, outperformed PT in terms of the overall average POR values.

Next, in Table 4.3, we display the average POR values produced by the implemented procedures at the end of Phase II for the two target value initialization methods, denoted by QUAD and PROJ as discussed in Section 5, as well as by using the VTVM-GPKC procedure by itself (initialized at the origin), for each class of test problems. Note that the Phase II runs of VTVM-GPKC were executed with an iteration

Table 4.2 POR Values after Phase I (%).

Prob	Perturbation Technique (PT)						Barrier-Lagrangian Dual Reformulation (BLR)					
	HS	PR	FR	SU	ADS	BFGS	HS	PR	FR	SU	ADS	BFGS
LP1	8.28	1.58	2.05	1.27	3.50	5.41	8.26	2.09	2.56	1.21	1.86	5.40
LP2	4.23	0.81	0.43	0.15	0.21	2.77	4.24	0.58	0.41	0.16	0.23	2.79
LP3	0.39	0.06	0.05	0.03	0.14	0.24	0.38	0.15	0.07	0.03	0.07	0.24
LP4	10.31	5.42	4.93	2.64	5.84	7.39	10.31	5.37	3.46	2.65	5.99	7.38
LP5	6.77	3.49	1.12	0.27	3.89	4.76	6.74	3.54	1.21	0.52	0.94	4.75
LP6	7.57	2.97	2.81	2.31	2.58	5.30	7.56	2.61	2.65	2.49	2.73	5.31
LP7	4.71	1.44	0.37	0.13	1.32	2.74	4.77	0.51	0.31	0.15	1.24	2.79
LP8	0.44	0.08	0.07	0.02	0.15	0.28	0.44	0.06	0.07	0.03	0.06	0.29
LP9	11.83	6.06	4.51	3.31	5.99	8.09	11.87	6.05	5.49	3.31	6.19	8.10
LP10	6.58	3.55	2.86	0.48	3.87	4.86	6.57	3.66	1.37	0.48	3.96	4.88
LP11	7.78	3.32	2.68	1.86	3.03	5.23	7.77	3.21	2.74	1.89	3.12	5.23
LP12	3.75	0.74	0.43	0.16	0.26	2.39	3.72	0.76	0.72	0.15	1.21	2.37
LP13	0.36	0.09	0.06	0.03	0.05	0.23	0.36	0.12	0.07	0.03	0.12	0.23
LP14	9.77	5.20	3.91	3.24	5.18	6.80	9.77	5.17	3.96	3.24	5.17	6.80
LP15	6.67	3.57	1.35	0.80	3.50	4.66	6.60	3.57	1.48	0.81	3.53	4.67
LP16	8.71	3.66	3.39	2.69	2.82	6.20	8.68	4.25	3.47	2.70	2.72	6.18
LP17	4.44	1.97	0.42	0.21	1.76	2.90	4.44	0.52	0.42	0.17	2.00	2.88
LP18	0.55	0.10	0.06	0.03	0.08	0.33	0.56	0.09	0.05	0.03	0.14	0.33
LP19	9.51	4.80	3.70	2.60	4.08	6.41	9.48	4.74	3.87	2.60	4.81	6.41
LP20	6.59	4.03	1.87	0.73	4.07	5.09	6.56	3.99	1.36	0.70	3.98	5.08
TR1	0.72	0.30	0.37	0.31	0.36	0.60	0.73	0.33	0.37	0.29	0.45	0.59
TR2	0.82	0.28	0.39	0.30	0.46	0.85	0.69	0.19	0.40	0.28	0.45	0.70
TR3	0.46	0.28	0.29	0.28	0.30	0.43	0.47	0.28	0.29	0.18	0.30	0.42
TR4	0.53	0.23	0.26	0.28	0.38	0.87	0.53	0.28	0.32	0.28	0.37	0.85
TR5	0.51	0.29	0.29	0.21	0.19	0.40	0.52	0.31	0.25	0.22	0.20	0.40
TR6	0.93	0.60	0.58	0.46	0.28	0.76	0.86	0.64	0.59	0.54	0.25	0.76
TR7	0.83	0.50	0.40	0.36	0.24	0.68	0.78	0.34	0.43	0.31	0.24	0.68
TR8	0.75	0.34	0.34	0.33	0.23	0.49	0.75	0.33	0.34	0.26	0.23	0.50
TR9	0.58	0.40	0.36	0.35	0.32	0.58	0.55	0.29	0.47	0.26	0.35	0.55
TR10	0.53	0.27	0.35	0.29	0.31	0.47	0.53	0.28	0.29	0.37	0.26	0.52
TR11	0.55	0.36	0.38	0.27	0.34	0.50	0.57	0.38	0.38	0.19	0.36	0.50
TR12	0.58	0.22	0.32	0.23	0.20	0.53	0.63	0.32	0.33	0.17	0.19	0.52
TR13	0.57	0.26	0.26	0.21	0.33	0.59	0.59	0.27	0.25	0.22	0.32	0.59
TR14	0.53	0.24	0.27	0.19	0.17	0.44	0.51	0.27	0.30	0.18	0.18	0.43
TR15	0.61	0.28	0.28	0.18	0.25	0.62	0.56	0.21	0.32	0.23	0.27	0.61
TR16	1.40	0.66	0.63	0.58	0.30	0.92	1.34	0.46	0.75	0.41	0.22	0.80
TR17	0.90	0.48	0.39	0.43	0.18	0.54	0.87	0.51	0.40	0.18	0.20	0.54
TR18	0.53	0.23	0.45	0.38	0.23	0.44	0.55	0.30	0.36	0.32	0.27	0.43
TR19	0.60	0.36	0.32	0.35	0.30	0.65	0.58	0.27	0.39	0.34	0.33	0.65
TR20	1.09	0.60	0.70	0.47	0.45	0.86	1.07	0.71	0.64	0.63	0.42	0.87
LP-Average	5.96	2.65	1.85	1.15	2.62	4.10	5.95	2.55	1.79	1.17	2.50	4.11
TR-Average	0.70	0.36	0.38	0.32	0.29	0.61	0.68	0.35	0.39	0.29	0.29	0.60
Overall Average	3.33	1.50	1.12	0.74	1.45	2.36	3.32	1.45	1.09	0.73	1.40	2.35

Table 4.3 Average POR Values after Phase II (%).

Average for	Phase I	Target Value Initialization Method	Strategy in Phase I						Stand-alone VTVM-GPKC
			HS	PR	FR	SU	ADS	BFGS	
LP	PT	QUAD	0.055	0.066	0.075	0.131	0.070	0.059	0.155
		PROJ	0.057	0.063	0.075	0.128	0.065	0.053	
	BLR	QUAD	0.053	0.067	0.076	0.114	0.074	0.055	
		PROJ	0.058	0.068	0.068	0.131	0.070	0.056	
TR	PT	QUAD	0.005	0.006	0.005	0.005	0.008	0.005	0.048
		PROJ	0.003	0.006	0.005	0.006	0.006	0.002	
	BLR	QUAD	0.004	0.005	0.005	0.007	0.008	0.004	
		PROJ	0.003	0.005	0.004	0.006	0.008	0.004	
Overall	PT	QUAD	0.030	0.036	0.040	0.068	0.039	0.032	0.102
		PROJ	0.030	0.035	0.040	0.067	0.035	0.028	
	BLR	QUAD	0.029	0.036	0.040	0.060	0.041	0.030	
		PROJ	0.030	0.036	0.036	0.069	0.039	0.030	

limit $k_{\max} = 1000$ (following the $K = 100$ iterations of Phase I), while for the stand-alone VTVM-GPKC runs, we used a larger iteration limit $k_{\max} = 2000$. Among the implemented procedures, the PT-BFGS combination for Phase I along with the PROJ target value initialization method yielded the best overall average POR value of 0.028% at the end of Phase II, while BLR-HS-QUAD was the second best combination, with an overall average POR value of 0.029%. We comment that although SU was the best strategy for the Phase I runs, it consistently revealed the worst overall performance over the two-phase runs. On the other hand, although the BFGS and HS strategies displayed relatively worse performances in Phase I, they yielded the best and the second best overall POR values among all combinations of Phase I techniques and target value initialization methods over the two-phase runs. This can be explained by the computational characteristic of the Phase II procedure in response to the quality and nondifferentiability of the initial solution. While one might generally presume that the performance of VTVM-GPKC can be improved by providing a better quality initial solution and initial target value via Phase I, however, we observed that if the initial solution is close to nondifferentiability, the Phase II procedure experiences difficulties in attaining ascents and in properly adjusting the target values. To verify this, let us define a *degree of nondifferentiability* (R) at π^K as

Table 4.4 Average Degrees of Nondifferentiability (*R*) at π^K.

Average for	Phase I	Strategy in Phase I					
		HS	PR	FR	SU	ADS	BFGS
LP	PT	11.10	8.95	7.34	6.16	8.80	9.93
	BLR	11.09	8.76	7.13	6.21	8.58	9.93
TR	PT	4.98	4.13	4.20	4.04	3.98	4.77
	BLR	4.94	4.10	4.24	3.95	3.98	4.75
Overall	PT	8.04	6.54	5.77	5.10	6.39	7.35
	BLR	8.02	6.43	5.68	5.08	6.28	7.34

$$R \equiv \frac{\sum_{j=1}^{n} \left| c_j - (\pi^K)^T A^j \right|}{n}.$$

Recall that if $\left| c_j - (\pi^K)^T A^j \right| = 0$ (or is small) for any $j = 1,...,n$, θ is (near-) nondifferentiable at π^K. The average R values computed for each combination of strategy and problem type are displayed in Table 4.4. Observe that the overall average R values when SU was employed are relatively small (5.10 and 5.08 for PT and BLR, respectively), as compared with the R values for the other strategies, particularly, for the more competitive methods HS and BFGS. Therefore, we conjecture that the performance of Phase II was affected by the degree of nondifferentiability as well as by the quality of the initial solution.

As far as the target value initialization method is concerned, PROJ revealed a similar or a slightly better overall performance for most combinations except for the BLR-HS and BLR-SU compositions, where the QUAD method yielded 0.001% and 0.009% smaller POR values than PROJ, respectively. Furthermore, PROJ slightly outperformed QUAD by attaining lesser overall average POR values to the extent ranging from 0.000-0.004% when PT was employed for Phase I. With regard to problem types, while the average POR values for QUAD and PROJ revealed mixed results for the LP problems, PROJ yielded smaller average POR values than those produced by QUAD by 0.000-0.002% when applied to TR, except for the PT-SU combination in which the average POR value of PROJ was 0.001% greater than that for QUAD.

It is important to note that as compared with the stand-alone VTVM-GPKC algorithm, the proposed two-phase procedures always revealed smaller overall average POR values, although the latter was run for only $K = 100$ and $k_{\max} = 1000$ iterations, while the former was run for $k_{\max} = 2000$ iterations. Even the maximum overall average POR value of 0.069% (BLR-SU- PROJ) was 0.033% less than that of VTVM, and we would expect the relative performance of the two-phase procedure to be even better when run for a comparable number of iterations. For example, we observed that the best combination (PT-BFGS-PROJ) yielded an overall average POR value of 0.0277% when run for 2000 iterations, which is 0.0001% smaller than that after 1000 iterations.

Next, the resulting average RCP values (see Section 3.5 for the definition of RCP) are displayed in Table 4.5. From Table 4.5, it is evident that the composition of Phase I via PT-ADS and Phase II initialized by the QUAD method yielded the best overall average RCP value of 3.19%. We remark here that this combination actually attained near-optimal values by the end of Phase I itself for 26 instances (6 out of 20 for LP and 20 out of 20 for TR). This procedure also yielded the smallest average RCP value (2.85%) for the class of TR problems. Furthermore, note that we attained near-optimal values for most TR instances during Phase I itself via PT, except for TR16 and TR20 when the HS strategy was employed. (This is the reason that, for the class of TR problems, the average RCP values for QUAD and PROJ in concert with PT were the same except for HS.) The best average RCP value of 3.13% for LP was obtained when the PT-FR-QUAD procedure was employed.

Also, observe that the stand-alone VTVM-GPKC procedure yielded no advantage over CPLEX 8.1 when solving our test bed of transportation problems, as evident from the average RCP value of 132.34%. However, even the maximum RCP value among the two-phase procedures was 33.64%, which corresponds to the composition of Phase I via BLR-HS and Phase II initialized by PROJ. Hence, the proposed two-phase procedures can effectively provide near-optimal values with a significant reduction in the effort required as compared with CPLEX 8.1. This can be greatly advantageous in LP-based branch-and-bound/cut methods for solving discrete optimization problems.

Noting that the SU strategy revealed the best performance for Phase I, we also experimented with a hybrid procedure in which we ran the PT-BFGS-PROJ combination

Table 4.5 Average RCP Values for Two-Phase Procedures (%).

Average for	Phase I	Target Value Initialization Method	Strategy in Phase I						MVTVM
			HS	PR	FR	SU	ADS	BFGS	
LP	PT	QUAD	3.94	3.48	3.13	3.14	3.53	4.32	5.54
		PROJ	4.36	4.13	3.71	3.53	3.95	4.72	
	BLR	QUAD	4.37	3.65	3.79	3.39	3.85	4.55	
		PROJ	4.69	4.26	4.12	3.94	4.43	5.13	
TR	PT	QUAD	10.23	3.58	3.83	4.05	2.85	6.40	132.34
		PROJ	15.92	3.58	3.83	4.05	2.85	6.40	
	BLR	QUAD	26.21	10.36	13.42	10.93	7.85	13.03	
		PROJ	33.64	10.36	13.42	10.93	7.85	13.03	
Overall	PT	QUAD	7.09	3.53	3.48	3.60	3.19	5.36	68.94
		PROJ	10.14	3.86	3.77	3.79	3.40	5.56	
	BLR	QUAD	15.29	7.01	8.60	7.16	5.85	8.79	
		PROJ	19.16	7.32	8.79	7.45	6.27	9.56	

in Phase I and then within Phase II, we reverted to executing 10 iterations of the Phase I procedure using the SU strategy whenever the target value was decreased (at Step 6). The average POR values for LP and TR obtained by this hybrid scheme were 0.062% and 0.006, respectively, while the overall average POR value was 0.034%. Note that this hybrid procedure yielded even worse performances than the procedure without incorporating any Phase I strategy. Again, we conjecture that this deterioration might be caused by the near-nondifferentiability of points produced by executing Phase I intermittently.

Based on the foregoing computational results, we conclude that the PT-BFGS-PROJ combination yielded the best POR value, while the composition of PT-ADS-QUAD produced the best RCP value among the implemented procedures. For the purpose of benchmarking, we also compared the POR and the RCP values obtained for the PT-BFGS-PROJ combination with those produced by the heuristic subgradient algorithms, HWC10 and HWC20, as described in Section 3.5. Table 4.6 displays the average POR and RCP values for these algorithms along with the PT-BFGS-PROJ composition that yielded the best solution quality over the two-phase runs. Note that both HWC10 and HWC20 could not attain near-optimal values for most TR problems (18 and 20 instances, respectively) and for four LP instances (LP4, 9, 14, and 19). To penalize these cases, we ascribed a corresponding RCP value of 100%. Note that despite assuming

Table 4.6 Benchmarking with HWC.

	Procedure	LP	TR	Overall
POR Values	PT-BFGS-PROJ	0.053	0.002	0.028
	HWC10	4.575	1.784	3.179
	HWC20	9.234	3.016	6.125
RCP Values	PT-BFGS-PROJ	4.718	6.402	5.560
	HWC10	30.297	99.465	64.881
	HWC20	33.694	100	66.847

a prior knowledge of a tight upper bound based on the optimal value, HWC10 yielded an 11436% larger overall average POR value than the PT-BFGS-PROJ combination, while its RCP value was 1167% larger than that of PT-BFGS-PROJ. It goes without saying that HWC20 displayed a far worse performance. Hence, based on these results, it appears that the two-phase procedure, PT-BFGS-PROJ, offers a much more effective methodology than the popular Held et al. scheme.

Finally, as commented in Remark 4.2 of Section 4.5, we experimented with the primal recovery schemes of Shor (1985) and Larsson et al. (1999) for the pure subgradient strategy and Sherali and Choi (1996) for the deflected subgradient strategy. (Let us denote these methods by SHOR, LPS, and SC, respectively.) For the sake of simplicity, we used optimal values instead of target values when computing step-lengths for SHOR and SC. Moreover, the parameter values for LPS were set as in their numerical experiments, and the deflection parameter for SC was set as 1. Each procedure was run for 2000 iterations with and without the BLR phase that provides a good starting point for ergodic primal recovery schemes. Table 4.7 displays the average *degree of primal infeasibility* (ρ) attained at the final resulting point $\tilde{x}$ (which effectively measures the near-optimality of this solution) defined as

$$\rho \equiv \frac{\sum_{j=1}^{m} \left| A^{(j)} \tilde{x} - b_j \right|}{m},$$

where $A^{(j)}$ is j^{th} row of A. For the class of LP problems, the SC scheme in conjunction with BLR yielded the smallest average ρ value of 0.47, while SHOR revealed the best average value of 0.83 for TR problems. Overall, SHOR implemented along with the BLR

Table 4.7 Average Degree of Primal Infeasibility.

Recovery Scheme	BLR	LP	TR	Overall
LPS	Without BLR	12.43	8.61	10.52
	With BLR	0.77	3.58	2.18
SHOR	Without BLR	4.30	2.34	3.32
	With BLR	0.53	0.83	0.68
SC	Without BLR	2.75	28.40	15.58
	With BLR	0.47	1.33	0.90

phase produced the smallest average ρ value of 0.68. Note that LPS, SHOR, and SC in conjunction with the BLR phase yielded overall average ρ values of 2.18, 0.68, and 0.90, respectively, which are only 20.7%, 20.5%, and 5.8% of the corresponding values produced by the same methods when not employing the BLR phase. Hence, we conclude that the BLR phase also significantly facilitates a more rapid convergence of ergodic type primal recovery schemes.

4.7 Summary and Conclusions

In this chapter, we designed two new approaches that attempt to circumvent or obviate nondifferentiability of the objective function, called the perturbation technique (PT) and the barrier-Lagrangian dual reformulation (BLR) method. In the PT procedure, whenever we encounter nondifferentiability, we utilize a perturbation to find a differentiable solution from which to continue the search. On the other hand, the BLR technique provides a differentiable Lagrangian dual function via a barrier problem, thereby assuring an ascent scheme. These techniques permit standard differentiable optimization methods to be utilized for solving NDO problems. Furthermore, the BLR procedure can be advantageously utilized to find a good starting primal solution to the Lagrangian subproblem at the prescribed initial dual solution in the context of various primal recovery schemes.

Based on these constructs, we proposed a two-phase algorithmic process. In the first phase (Phase I) of this method, PT or BLR are utilized to discover a relatively good solution for offering a warm-start to Phase II. Consequently, a theoretically convergent

NDO procedure, VTVM-GPKC recommended in Chapter 3, is employed for the second phase. Forty transportation and general linear programming test problems having various sizes and degrees of degeneracy were randomly generated for our computational study. We also experimented with six gradient deflection strategies including HS, PR, FR, SU, ADS, and BFGS for Phase I, and two target value initialization methods QUAD and PROJ for Phase II. Based on our computational results, both PT and BLR of Phase I helped significantly improve the algorithmic performance as compared with implementing VTVM-GPKC alone, even though the latter was run for twice as many iterations as the two-phase procedures. Among the gradient deflection direction-finding strategies, SU revealed the best performance for Phase I. However, using the BFGS and HS strategies in Phase I yielded the best and second best overall performance, respectively, for the two-phase method. This phenomenon was explained by the degree of nondifferentiability of the initial solution fed into Phase II. Between the two proposed target value initialization methods, PROJ revealed a better overall performance than did QUAD for most combinations of Phase I methods and direction-finding strategies. Among all combinations, PT-BFGS-PROJ yielded the best overall quality solutions, producing an average POR value of 0.028%, which measures the relative final optimality gap to the initial optimality gap.

As far as CPU time is concerned, the combination of PT-ADS-QUAD attained near-optimal values (having POR values lesser than or equal to 1%) with the least effort in our experiments. The overall average RCP value produced by this combination, which measures the CPU effort as a percentage of the time required by the commercial package CPLEX 8.1 to produce the same quality solution, was 3.19%. That is, PT-ADS-QUAD required only 3.19% of the computational time consumed by CPLEX 8.1 to produce the same near-optimal values. Furthermore, the overall average RCP value of the PT-BFGS-PROJ combination, which produced the best quality solutions on average, was 5.56%. We also experimented with a widely used subgradient heuristic of Held et al. (1974), which yielded an 11436% larger average POR value and an 1167% larger RCP value than the corresponding values produced by the PT-BFGS-PROJ combination. Finally, we tested some ergodic primal recovery strategies with and without employing BLR as a

warm-start, and observed that an initial BLR phase can significantly enhance the convergence of such primal recovery schemes.

Chapter 5:
A New Variable Target Value Framework:
Trust Region Target Value Method

5.1 Introduction

The VTVM algorithm may experience an adverse crawling phenomenon for some classes of problems as cited in Chapter 3. Although certain modifications such as applying acceleration factors to η and performing quadratic-fit line-searches can alleviate this drawback as discussed in Chapter 3, there still exists a need for improvement. More specifically, a particular shortcoming of concern here is that the VTVM procedure lacks robustness because it requires the fine-tuning of several parameter values for different classes of problems in order to improve its performance. Although these parameter settings are determined based on intensive computational experience, they may not work well for all types of problems, as we observed with set partitioning problems in Chapter 3. Accordingly, in this chapter, we develop a new variable target value method, called the *Trust Region Target Value* (TRTV) method, which entails the management of only a few parameters.

In Section 5.2, we present the details of the proposed TRTV method. Then, in Section 5.3, we provide a convergence analysis when the PS, ADS, and VA direction strategies are employed in concert with TRTV. Moreover, variations of TRTV are discussed in Section 5.4. A relevant computational study is presented in Section 5.5, which is followed by concluding remarks in Section 5.6.

5.2 Trust Region Target Value Method

Consider an equality-constrained linear program and its Lagrangian dual formulation respectively given in the form:

LP: Minimize $c^T x$ (5.1a)

subject to $Ax = b$ (5.1b)

$$x \in X \equiv \{x \in R^n : Dx \geq d, 0 \leq x \leq u\}, \quad (5.1c)$$

and

LD: Maximize $\theta(\pi)$ (5.2a)

where $\theta(\pi) = \min\{c^T x + \pi^T (b - Ax) : x \in X\}$. (5.2b)

Note that the dual vector $\pi \in R^m$ in (5.2) is unrestricted. Let $\bar{\pi}$ be some arbitrary dual solution in R^m, and suppose that we have a trust region

$$\Pi = \{\pi \in R^m : \bar{\pi} - \mu e \leq \pi \leq \bar{\pi} + \mu e\}$$

that contains an optimal solution π^* to LD of (5.2), where $\varepsilon \in R^m$ is a vector of ones. Then, exchanging the max and min operators and noting the linearity and boundedness of X, and the linearity of the Lagrangian function in each of the primal and dual variable spaces, we have the following holding true. (In general, $\max_{v \in V} \min_{w \in W} f(v, w) = \min_{w \in W} \max_{v \in V} f(v, w)$ when, for nonempty closed convex sets $V \in R^m$ and $W \in R^n$, f is a continuous finite concave-convex function on $V \times W$, and at least one of V and W is bounded (see Rockafellar, 1970).)

$$\theta^* = \max_{\pi \in \Pi} \min_{x \in X} \{c^T x + \pi^T (b - Ax)\} \quad (5.3a)$$

$$= \min_{x \in X} \max_{\pi \in \Pi} \{c^T x + \pi^T (b - Ax)\} \quad (5.3b)$$

$$= \min_{x \in X} \phi(x), \text{ say,} \quad (5.3c)$$

where

$$\phi(x) \equiv \max_{\pi \in \Pi} \{c^T x + \pi^T (b - Ax)\}. \quad (5.3d)$$

The motivation behind our TRTV method is that if the trust region Π contains

π^*, we have that $\phi(x) \geq \theta^*$, $\forall x \in X$, and hence, $\phi(x)$, for any $x \in X$, would yield an upper bound for θ^*. Noting that the incumbent dual objective value is a lower bound for θ^*, we can estimate the target value via an average of these two bounds. The algorithm starts with an initial target value estimated by an average of $\theta_1 = \theta(\pi^1)$ for some π^1 and $\phi(x^1)$, where x^1 solves $\theta(\pi^1)$, along with a trust region centered at π^1. Then, at iteration k, the target value is set to an average of the incumbent dual objective value z_k and $\phi(\bar{x}^k)$, where $\bar{x}^k$ is the solution to the Lagrangian dual subproblem at the incumbent point. By employing a pure or deflected subgradient direction strategy while continually centering Π at the incumbent solution, we eventually get dual convergence with π^* appearing at the center of the trust region. Moreover, the trust region is itself reduced to this optimal point π^* by decreasing μ in the process as the number of iterations increases, so that we obtain $\{\phi(\bar{x}^k)\} \rightarrow \theta^*$.

To further develop such a target value and trust region update scheme, suppose that the algorithm proceeds through $\bar{\gamma}$ consecutive iterations without experiencing a sufficient improvement. Then, it is suspected that while the incumbent solution might be close to π^*, the trust region is too large to induce an improvement. Therefore, we reduce the trust region parameter μ in this case, subject to a related convergence condition. Also, whenever the dual incumbent solution is updated, the target value tends to increase due to the larger lower bound, and hence, a certain step-length level is retained in subsequent iterations. This obviates a crawling phenomenon that tends to creep into the VTVM procedure and the Level Algorithm. The details of such a TRTV method can be described as follows.

TRTV Algorithm

Initialization Step: Select a termination tolerance $0 < \varepsilon_0 < 1$ on subgradient norms and $\bar{\varepsilon} \geq 0$ on trust region sizes, an iteration limit $k_{\max}$, a step-length parameter $\beta \in (0,1]$, a sufficient improvement parameter $\sigma \in [\varepsilon_0, 1)$, and a threshold value $\bar{\gamma}$ for decreasing the trust region parameter. Let π^1 be an arbitrary initial dual vector, and compute g^1, x^1,

and θ_1. Set incumbent vectors and the incumbent dual objective value as $\bar{\pi}^1 = \pi^1$, $\bar{g}^1 = g^1$, $\bar{x}^1 = x^1$, and $z_1 = \theta_1$. If $\|g^1\| < \varepsilon_0$, terminate the algorithm. Else, define index sets $I_+ \equiv \{i : b_i > A^{(i)}x^1\}$ and $I_- \equiv \{i : b_i < A^{(i)}x^1\}$, where $A^{(i)}$ denotes the i^{th} row of A. Solve the following knapsack problem.

$$\text{Minimize} \quad c^T x \tag{5.4a}$$

$$\text{subject to} \quad v^T A x = v^T b \tag{5.4b}$$

$$x \in X, \tag{5.4c}$$

where

$$v_i = \begin{cases} 1 & \text{if } i \in I_+ \\ -1 & \text{if } i \in I_- \\ 0 & \text{otherwise.} \end{cases} \tag{5.4d}$$

Let μ_1 be the absolute value of the optimal dual solution associated with the constraint in (5.4b). Calculate $\phi(\bar{x}^1)$ with the trust region given by $\Pi = \{\pi : \bar{\pi}^1 - \mu_1 e \le \pi \le \bar{\pi}^1 + \mu_1 e\}$, and initialize the target value as $w_1 = (z_1 + \phi(\bar{x}^1))/2$. Let the sufficient improvement threshold value $\overline{w}_1 \equiv z_1 + \sigma(w_1 - z_1)$. Initialize the target value adjustment counter $l = 1$, the overall iteration counter $k = 1$, the iteration counter for consecutive non-improvements $\gamma = 0$, and the reset indicator RESET=1.

Step 1: Call a subroutine that implements a direction-finding and step-length strategy to obtain π^{k+1}. Increment $k \leftarrow k+1$ and $\gamma \leftarrow \gamma + 1$. Compute θ_k and $g^k \in \partial\theta(\pi^k)$. Put RESET=0.

Step 2: If $\theta_k > z_{k-1}$, set $z_k = \theta_k$, $\bar{\pi}^k = \pi^k$, $\bar{g}^k = g^k$, $\bar{x}^k = x^k$, and proceed to Step 3. Otherwise, put $z_k = z_{k-1}$, $\bar{\pi}^k = \bar{\pi}^{k-1}$, $\bar{g}^k = \bar{g}^{k-1}$, and $\bar{x}^k = \bar{x}^{k-1}$. If $\gamma \le \bar{\gamma}$, return to Step 1. Else, go to Step 4.

Step 3: If $\|g^k\| < \varepsilon_0$ or $k \ge k_{\max}$, terminate the algorithm. If $\theta_k < \overline{w}_l$, then return to Step 1 if $\gamma \le \bar{\gamma}$; else go to Step 4. If $\theta_k \ge \overline{w}_l$, then update $\mu_{l+1} = \mu_l$, and go to Step 5.

Step 4: Put $\mu_{l+1} = (\mu_1 \mu_l)/(\mu_1 + \mu_l)$. Terminate the algorithm if $\mu_{l+1} \le \bar{\varepsilon}$. Else, replace $\pi^k \leftarrow \bar{\pi}^k$, $g^k \leftarrow \bar{g}^k$, and $\theta_k \leftarrow z_k$, set RESET=1, and proceed to Step 5.

Step 5: (Outer Iteration) Increment $l \leftarrow l+1$. Restructure the trust region as $\Pi = \{\pi : \bar{\pi}^k - \mu_l e \le \pi \le \bar{\pi}^k + \mu_l e\}$. Obtain a new target value via $w_l = (z_k + \phi(\bar{x}^k))/2$, update $\bar{w}_l = z_k + \sigma(w_l - z_k)$, and reset $\gamma = 0$. Return to Step 1.

Remark 5.1: For our computational study, we set $\varepsilon_0 = 10^{-6}$, $\bar{\varepsilon} = 0$, $k_{\max} = 2000$, $\beta = 0.8$, and $\sigma = 0.5$. Also, the threshold iteration number $\bar{\gamma}$ is initialized at 10 and updated according to $\min\{50, \bar{\gamma}+10\}$ whenever the value of μ_l is decreased at Step 4. Furthermore, in order to facilitate a quick adjustment of target values in earlier stages, we halve the trust region parameter μ_l in Step 4 until the threshold value $\max\{1, 0.001\mu_1\}$ is attained. Thereafter, for the sake of convergence, we reduce this parameter using a harmonic series as prescribed in Step 4 of the algorithm.

5.3 Convergence of the TRTV Algorithm

Assume that the TRTV algorithm, run without the iteration limit $k_{\max}$ and $\bar{\varepsilon} = 0$, generates an infinite sequence such that $\{\bar{\pi}^k\} \to \bar{\pi}^\infty$. Then, the convergence arguments for TRTV in concert with a conjugate subgradient strategy can be derived as follows.

Lemma 5.1

Consider the TRTV algorithm run with $k_{\max} = \infty$ and $\bar{\varepsilon} = 0$, and suppose that an infinite sequence of incumbent iterates $\{\bar{\pi}^k\}$ is generated. Assume that there exists an optimum π^* to Problem LD such that $\theta(\pi^*) < \infty$. Then, the trust region parameter μ_l satisfies:

i) $\{\mu_l\} \to 0$,

ii) $\sum_{l=1}^{\infty} \mu_l = \infty$.

Proof: Suppose that $\{\mu_l\} \not\to 0$, so that by the steps of the algorithm, we then have $\mu_l = \mu_\infty > 0$ for $l \geq L$. Consider a target value update at the iteration $k(l)$, where $k(l)$ is the iteration corresponding to an outer iteration $l \geq L$. Note that a solution to $\phi(\bar{x}^{k(l)})$, denoted by $\tilde{\pi}^l$, is an extreme point of $\Pi_l = \{\pi : \bar{\pi}^{k(l)} - \mu_\infty e \leq \pi \leq \bar{\pi}^{k(l)} + \mu_\infty e\}$. Furthermore, note that

$$\begin{aligned}\phi(\bar{x}^{k(l)}) &= \max_{\pi \in \Pi_l} \{c^T \bar{x}^{k(l)} + \pi^T \bar{g}^{k(l)}\} \\ &= \max_{\pi \in \Pi_l} \{\theta(\bar{\pi}^{k(l)}) + (\pi - \bar{\pi}^{k(l)})^T \bar{g}^{k(l)}\} \\ &= \theta(\bar{\pi}^{k(l)}) + (\tilde{\pi}^l - \bar{\pi}^{k(l)})^T \bar{g}^{k(l)}. \end{aligned} \tag{5.5}$$

Hence, $\tilde{\pi}^l$ maximizes the linear approximation of θ at $\bar{\pi}^{k(l)}$ within the trust region Π_l. Noting that $\bar{\pi}^{k(l)} + \mu_\infty \bar{g}^{k(l)} \Big/ \left\| \bar{g}^{k(l)} \right\| \in \Pi_l$, we have from (5.5) that

$$\phi(\bar{x}^{k(l)}) \geq \theta(\bar{\pi}^{k(l)}) + \mu_\infty \left\| g^{k(l)} \right\| = z_{k(l)} + \mu_\infty \left\| g^{k(l)} \right\| \geq z_{k(l)} + \mu_\infty \varepsilon_0. \tag{5.6}$$

From Step 5 of the algorithm, (5.6), and $\sigma \geq \varepsilon_0$, we have

$$\bar{w}_{l+1} - z_{k(l)} = \sigma(w_{l+1} - z_{k(l)}) = \sigma \frac{\phi(\bar{x}^{k(l)}) - z_{k(l)}}{2} \geq \frac{\sigma \mu_\infty \varepsilon_0}{2} \geq \frac{\mu_\infty \varepsilon_0^2}{2} > 0. \tag{5.7}$$

Let us denote an attained improvement in θ between the outer iterations l and $l+1$ by Δ_l. Then, from (5.7), by summing the achieved improvements over the outer iterations, we get

$$\sum_{l=1}^{\infty} \Delta_l \geq \sum_{l=L}^{\infty} \Delta_l \geq \sum_{l=L}^{\infty} (\bar{w}_{l+1} - z_{k(l)}) \geq \sum_{l=L}^{\infty} \frac{\mu_\infty \varepsilon_0^2}{2} = \infty. \tag{5.8}$$

Since $\left|\theta(\pi^1)\right| < \infty$ and $\theta(\pi^*) < \infty$, (5.8) leads to a contradiction. Therefore, $\{\mu_l\} \to 0$ holds true. Now, note that μ_l is decreased according to a harmonic sequence with $\mu_l \geq \mu_1 / l$, $\forall l$, if no sufficient improvement achieved, or remains the same otherwise. Since the harmonic series diverges to ∞, we have

$$\sum_{l=1}^{\infty} \mu_l \geq \mu_1 \sum_{l=1}^{\infty} \frac{1}{l} = \infty.$$

Hence, $\sum_{l=1}^{\infty} \mu_l = \infty$ holds true as well. ■

Theorem 5.1

Consider the TRTV algorithm applied by using a conjugate subgradient direction-finding strategy at Step 1 as given by $d^{k+1} = g^k + \alpha_k d^{k-1}$, where $\alpha_k \geq 0$, and with a prescribed step-length $\lambda_k = \beta[w_l - \theta_k] \Big/ \left\| d^k \right\|^2$, where $\beta \in (0,1]$. Suppose that an infinite sequence $\{\pi^k\}$ is generated, where $\left\{ \left\| d^k \right\| \right\}$ is bounded, and $\left\| d^k \right\| \geq \varepsilon_d > 0 \quad \forall k$. Then, we have $\{z_k\} \to \theta^*$.

Proof: Consider the iteration $k(l)$ corresponding to an outer iteration l. Noting that $\{z_k\}$ is convergent since it is monotone and bounded, let us denote its limit value as z_∞. First, following a derivation similar to (5.6), note that we have

$$\phi(\bar{x}^{k(l)}) - z_{k(l)} \geq \mu_l \left\| g^{k(l)} \right\| \geq \mu_l \varepsilon_0. \tag{5.9}$$

From Lemma 5.1, adding both sides of (5.9) yields

$$\sum_{l=1}^{\infty} (\phi(\bar{x}^{k(l)}) - z_{k(l)}) \geq \varepsilon_0 \sum_{l=1}^{\infty} \mu_l = \infty.$$

Because $\theta_k \le z_k$, $w_l = (\phi(x^{k(l)}) + z_{k(l)})/2$, and $\|d^k\|$ is bounded, the summation of step-lengths yields

$$\begin{aligned}\sum_{k=1}^{\infty} \lambda_k &= \beta \sum_{k=1}^{\infty} \frac{w_l - \theta_k}{\|d^k\|^2} \ge \beta \sum_{k=1}^{\infty} \frac{w_l - z_k}{\|d^k\|^2} \ge \beta \sum_{l=1}^{\infty} \frac{w_l - z_{k(l)}}{\|d^{k(l)}\|^2} \\ &\ge \frac{\beta}{2M^2} \sum_{l=1}^{\infty} (\phi(\bar{x}^{k(l)}) - z_{k(l)}) = \infty,\end{aligned} \tag{5.10}$$

where $M \equiv \sup\{\|d^k\|\}$. Notice that since $\{\mu_l\} \to 0$ from Lemma 5.1, we have

$$\{\phi(\bar{x}^{k(l)})\} \to z_\infty \text{ as } l \to \infty,$$

and hence,

$$\{w_l\} \to z_\infty \text{ as } l \to \infty. \tag{5.11}$$

Now, suppose that, as $k \to \infty$,

$$\{z_k\} \to z_\infty < \theta^*. \tag{5.12}$$

From (5.11) and (5.12), we have a positive integer L such that $w_l \le \theta^* - \rho$ for $l \ge L$, where $\rho \in (0, \theta^* - z_\infty)$. Note that we have $(\pi^k - \pi^*)^T d^{k-1} \le 0$, $\forall k \ge 2$ following Lemma 1 of Sherali et al. (2000). Therefore, from $\alpha_k \ge 0$, the concavity of θ, and $w_l < \theta^*$, we get, for $k \ge k(L)$,

$$\begin{aligned}\|\pi^{k+1} - \pi^*\|^2 &= \|\pi^k + \lambda_k d^k - \pi^*\|^2 \\ &= \|\pi^k - \pi^*\|^2 + \lambda_k^2 \|d^k\|^2 + 2\lambda_k (\pi^k - \pi^*)^T d^k\end{aligned}$$

$$= \left\|\pi^k - \pi^*\right\|^2 + \beta^2 \frac{(w_l - \theta_k)^2}{\left\|d^k\right\|^2} + 2\beta \frac{(w_l - \theta_k)}{\left\|d^k\right\|}(\pi^k - \pi^*)^T (g^k + \alpha_k d^{k-1})$$

$$\le \left\|\pi^k - \pi^*\right\|^2 + \beta^2 \frac{(w_l - \theta_k)^2}{\left\|d^k\right\|^2} + 2\beta \frac{(w_l - \theta_k)}{\left\|d^k\right\|}(\pi^k - \pi^*)^T g^k$$

$$\le \left\|\pi^k - \pi^*\right\|^2 + \beta^2 \frac{(w_l - \theta_k)^2}{\left\|d^k\right\|^2} - 2\beta \frac{(w_l - \theta_k)(\theta^* - \theta_k)}{\left\|d^k\right\|}$$

$$\le \left\|\pi^k - \pi^*\right\|^2 - \beta(2-\beta) \frac{(w_l - \theta_k)^2}{\left\|d^k\right\|^2}.$$

Since $\left\{\left\|\pi^k - \pi^*\right\|\right\}^2$ is monotone decreasing, and being bounded below, we must have $\{(w_l - \theta_k)\} \to 0$, and in turn,

$$\{\lambda_k\} \to 0.$$

Note that, since θ is a continuous function and $z_\infty < \theta^*$, we have a level set $\Lambda(\theta^* - \rho) \equiv \{\pi : \theta(\pi) \ge \theta^* - \rho\}$ such that $\operatorname{int} \Lambda(z_\infty) \ne \varnothing$. Following Polyak's (1967) convergence argument, let us consider $\tilde{\pi} \in \operatorname{int} \Lambda(\theta^* - \rho)$ and choose $\delta > 0$ such that $N_\delta(\tilde{\pi}) \equiv \left\{\pi : \left\|\pi - \tilde{\pi}\right\| \le \delta\right\} \subset \Lambda(\theta^* - \rho)$. Furthermore, consider the subsequence of outer iterations $\{\bar{l}_i\}$ for $i = 1, 2, \ldots$, with $\bar{l}_1 \ge L$, that is triggered by Step 4 with resetting, where the $\bar{l}_i$ values correspond to those at the end of the subsequent Step 5 updates. Given any such $\bar{l}_i$, let us inductively show that for $\hat{\pi} \in N_\delta(\tilde{\pi})$, we have

$$(\pi^k - \hat{\pi})^T d^{k-1} \le 0 \qquad \text{for } k(\bar{l}_i) + 1 \le k \le k(\bar{l}_{i+1}) - 1. \tag{5.13}$$

For $k = k(\bar{l}_i) + 1$, since $d^{k(\bar{l}_i)} = g^{k(\bar{l}_i)}$, θ is concave, $w_{\bar{l}_i} < \theta(\hat{\pi})$, $\beta \in (0, 1]$, and

$\theta(\hat{\pi}) \geq \theta_{k(\bar{l}_i)}$, we have,

$$\begin{aligned}(\pi^{k(\bar{l}_i)+1} - \hat{\pi})^T d^{k(\bar{l}_i)} &= (\pi^{k(\bar{l}_i)} + \lambda_{k(\bar{l}_i)} g^{k(\bar{l}_i)} - \hat{\pi})^T g^{k(\bar{l}_i)} \\ &= (\pi^{k(\bar{l}_i)} - \hat{\pi})^T g^{k(\bar{l}_i)} + \beta(w_{\bar{l}_i} - \theta_{k(\bar{l}_i)}) \\ &\leq -(\theta(\hat{\pi}) - \theta_{k(\bar{l}_i)}) + \beta(w_{\bar{l}_i} - \theta_{k(\bar{l}_i)}) \\ &\leq (\beta - 1)(\theta(\hat{\pi}) - \theta_{k(\bar{l}_i)}) \leq 0 .\end{aligned}$$

Now, assume that the inequality in (5.13) holds true for $k(\bar{l}_i)+1,\ldots,k(\bar{l}_i)+j-1$, and consider $k = k(\bar{l}_i) + j$ such that $2 \leq j < k(\bar{l}_{i+1}) - k(\bar{l}_i)$. Then, from $w_{\bar{l}_i} < \theta(\hat{\pi})$, $\alpha_k \geq 0$, the induction hypothesis, and the concavity of θ, we get

$$\begin{aligned}(\pi^{k(\bar{l}_i)+j} - \hat{\pi})^T d^{k(\bar{l}_i)+j-1} &= (\pi^{k(\bar{l}_i)+j-1} + \lambda_{k(\bar{l}_i)+j-1} d^{k(\bar{l}_i)+j-1} - \hat{\pi})^T d^{k(\bar{l}_i)+j-1} \\ &= (\pi^{k(\bar{l}_i)+j-1} - \hat{\pi})^T d^{k(\bar{l}_i)+j-1} + \beta(w_{\bar{l}_i} - \theta_{k(\bar{l}_i)+j-1}) \\ &< (\pi^{k(\bar{l}_i)+j-1} - \hat{\pi})^T d^{k(\bar{l}_i)+j-1} + \beta(\theta(\hat{\pi}) - \theta_{k(\bar{l}_i)+j-1}) \\ &= (\pi^{k(\bar{l}_i)+j-1} - \hat{\pi})^T (g^{k(\bar{l}_i)+j-1} + \alpha_{k(\bar{l}_i)+j-2} d^{k(\bar{l}_i)+j-2}) + \beta(\theta(\hat{\pi}) - \theta_{k(\bar{l}_i)+j-1}) \\ &\leq -(\theta(\hat{\pi}) - \theta_{k(\bar{l}_i)+j-1}) + \beta(\theta(\hat{\pi}) - \theta_{k(\bar{l}_i)+j-1}) \\ &= (\beta - 1)(\theta(\hat{\pi}) - \theta_{k(\bar{l}_i)+j-1}) \leq 0 .\end{aligned}$$

Hence, the inequality in (5.13) holds true for $k = k(\bar{l}_i)+1,\ldots,k(\bar{l}_{i+1})-1$, $\forall \bar{l}_i$. From (5.13), the concavity of θ, and $\theta(\hat{\pi}) \geq \theta_k$, we get, for $k = k(\bar{l}_i)+1,\ldots,k(\bar{l}_{i+1})-1$, $\forall \bar{l}_i$,

$$\begin{aligned}(\pi^k - \hat{\pi})^T d^k &= (\pi^k - \hat{\pi})^T (g^k + \alpha_{k-1} d^{k-1}) \\ &\leq (\pi^k - \hat{\pi})^T g^k\end{aligned}$$

$$\leq \theta_k - \theta(\hat{\pi}) \leq 0 \,. \tag{5.14}$$

Furthermore, note that because $d^{k(\bar{l}_i)} = g^{k(\bar{l}_i)}$, we have

$$(\pi^k - \hat{\pi})^T d^k = (\pi^k - \hat{\pi})^T g^k \leq \theta_k - \theta(\hat{\pi}) \leq 0 \text{ for } k = \bar{l}_i \,,\ \forall \bar{l}_i \,. \tag{5.15}$$

Therefore, from (5.14) and (5.15), we have

$$(\pi^k - \hat{\pi})^T d^k \leq 0 \,, \qquad \forall k \geq k(\bar{l}_i) \,. \tag{5.16}$$

Thus, using (5.16) with $\hat{\pi} \equiv \tilde{\pi} - \delta\, d^k \big/ \left\| d^k \right\| \in N_\delta(\tilde{\pi})$, for any $k \geq k(\bar{l}_i)$, we get

$$\begin{aligned}
\left\| \pi^{k+1} - \tilde{\pi} \right\|^2 &= \left\| \pi^k + \lambda_k d^k - \tilde{\pi} \right\|^2 \\
&= \left\| \pi^k - \tilde{\pi} \right\|^2 + \lambda_k^2 \left\| d^k \right\|^2 + 2\lambda_k (\pi^k - \tilde{\pi})^T d^k \\
&= \left\| \pi^k - \tilde{\pi} \right\|^2 + \lambda_k^2 \left\| d^k \right\|^2 + 2\lambda_k \left[\pi^k - \left(\tilde{\pi} - \delta\, d^k \big/ \left\| d^k \right\| \right) \right]^T d^k - 2\delta\lambda_k \left\| d^k \right\| \\
&\leq \left\| \pi^k - \tilde{\pi} \right\|^2 + \lambda_k^2 \left\| d^k \right\|^2 - 2\delta\lambda_k \left\| d^k \right\| .
\end{aligned} \tag{5.17}$$

Since $\{\lambda_k\} \to 0$ and $\left\| d^k \right\| \leq M$, consider a positive integer $K \geq k(\bar{l}_1)$ such that $\lambda_k \left\| d^k \right\| \leq \delta$ for $k \geq K$. The summation of both sides in (5.17) from K to $K + p$ gives

$$\begin{aligned}
0 \ &\leq \left\| \pi^{K+p+1} - \tilde{\pi} \right\|^2 \leq \left\| \pi^K - \tilde{\pi} \right\|^2 + \sum_{j=K}^{K+p} \lambda_j \left(\lambda_j \left\| d^j \right\| - 2\delta \right) \left\| d^j \right\| \\
&\leq \left\| \pi^K - \tilde{\pi} \right\|^2 - \delta \sum_{j=K}^{K+p} \lambda_j \left\| d^j \right\|
\end{aligned}$$

$$\leq \left\| \pi^K - \tilde{\pi} \right\|^2 - \delta \varepsilon_d \sum_{j=K}^{K+p} \lambda_j \,. \tag{5.18}$$

Note that, as $p \to \infty$, the inequality in (5.18) contradicts that $\sum_{k=1}^{\infty} \lambda_k = \infty$ in (5.1). Therefore, we must have $z_\infty = \theta^*$. ■

Corollary 5.1.1

Consider the TRTV algorithm as described in Theorem 5.1, along with the PS strategy with $\beta \in (0,1]$. Then, we have $\{z_k\} \to \theta^*$.

Proof: Letting the deflection parameter α_k be zero, the proof is identical to that for Theorem 5.1. ■

Corollary 5.1.2

Consider the TRTV algorithm as described in Theorem 5.1, along with the VA strategy with the step-length parameter β and the geometric moving average parameter α satisfying $0 < \beta \leq \alpha \leq 1$. Then, we have $\{z_k\} \to \theta^*$.

Proof: Recall that, at iteration k, a search direction is prescribed as $d^k = \alpha g^k + (1-\alpha) d^{k-1}$ in the VA strategy. Hence, given an iterate π^k, the VA strategy produces the next iterate π^{k+1} as

$$\begin{aligned}
\pi^{k+1} &= \pi^k + \lambda_k d^k \\
&= \pi^k + \beta \frac{(w_l - \theta_k)}{\left\| \alpha g^k + (1-\alpha) d^{k-1} \right\|^2} [\alpha g^k + (1-\alpha) d^{k-1}] \\
&= \pi^k + \beta \frac{(w_l - \theta_k)}{\alpha^2 \left\| g^k + \frac{(1-\alpha)}{\alpha} d^{k-1} \right\|^2} \alpha \left[g^k + \frac{(1-\alpha)}{\alpha} d^{k-1} \right]
\end{aligned}$$

$$= \pi^k + \left(\frac{\beta}{\alpha}\right) \frac{(w_l - \theta_k)}{\left\| g^k + \frac{(1-\alpha)}{\alpha} d^{k-1} \right\|^2} \left[g^k + \frac{(1-\alpha)}{\alpha} d^{k-1} \right]. \qquad (5.19)$$

Letting $\beta' \equiv \beta/\alpha$, (5.19) reduces to a deflected subgradient strategy, where the deflection parameter α_k is equal to $(1-\alpha)/\alpha$. Furthermore, since $\beta \le \alpha$, we have $\beta' \in (0, 1]$. Therefore, the convergence directly follows from Theorem 5.1. ■

5.4 Variations of the TRTV Algorithm

Since we are adopting a hypercube trust region, scaling the constraints can be beneficial due to the normalizing effect this would have on the dual variables. One way to accommodate this observation without performing an actual scaling is to adopt a more general hyperrectangular trust region. Dualizing the constraints in (5.1b), yields the following i^{th} term in the objective function:

$$\pi_i (b_i - A^{(i)} x) = \left\| (b_i, A^{(i)}) \right\| \pi_i \left(\frac{b_i}{\left\| (b_i, A^{(i)}) \right\|} - \frac{A^{(i)} x}{\left\| (b_i, A^{(i)}) \right\|} \right) \equiv \pi_i' \left(\frac{b_i}{\left\| (b_i, A^{(i)}) \right\|} - \frac{A^{(i)} x}{\left\| (b_i, A^{(i)}) \right\|} \right).$$

Assuming that the scaled dual variable values π_i' are close to each other, and thus the hypercube trust region is appropriate for the π_i'-variables, the trust range for each original dual variable π_i can be selected to be proportional to the reciprocal of $\left\| (b_i, A^{(i)}) \right\|$. Accordingly, instead of using μe for defining the trust region Π, we could define a trust region vector t, where $t_i = \left\| (b_i, A^{(i)}) \right\|^{-1}$, and then use

$$\Pi = \{ \pi \ge 0 : \bar{\pi} - \mu t \le \pi \le \bar{\pi} + \mu t \}.$$

Table 5.1 Solution Quality Comparison for TRTV, VTVM-GPKC, and HWCs.

Employed Procedure		Average POR Value (%)			Standard Deviation		
		LP	TR	Overall	LP	TR	Overall
TRTV	PS	0.050	0.012	0.031	0.055	0.003	0.043
	VA	0.061	0.015	0.038	0.054	0.005	0.045
	ADS	0.068	0.029	0.049	0.048	0.009	0.040
	GPKC	0.072	0.010	0.041	0.068	0.004	0.057
VTVM-GPKC		0.155	0.048	0.102	0.303	0.058	0.222
HWC10		4.575	1.784	3.179	9.482	0.790	6.790
HWC20		9.234	3.016	6.125	19.034	0.602	13.660

5.5 Computational Study

The TRTV procedure was implemented on the same set of test problems as in the Chapter 4. Noting that these test problems were generated within scaled ranges, we did not experiment with the scaling method proposed in the previous section. For solving the knapsack problem to initialize the target value, we used the CPLEX callable library within the procedure. We experimented with four direction-finding strategies: PS, VA, ADS, and GPKC. The average POR values (see Section 2.5) for each class of test problems are displayed in Table 5.1 along with those of HWC10, HWC20, and VTVM-GPKC. We also employed the projected quadratic-fit line-search for the TRTV algorithm, since we observed that this played a beneficial role in the computational study in Chapter 3. The best overall average POR value of 0.031% was obtained when the TRTV procedure was employed in concert with PS, while the TRTV-VA combination yielded the second best value of 0.038%. Note that the TRTV algorithm consistently outperformed the other procedures. For example, even the largest overall average POR value produced by TRTV (0.049%) was yet 0.053%, 3.131%, and 6.076% lesser than the corresponding values for VTVM-GPKC, HWC10, and HWC20, respectively. When the TRTV algorithm was implemented for solving LP problems, the rank-order of the employed strategies in terms of the average POR values was PS, VA, ADS, and GPKC from the best (0.050%) to the worst (0.072%). However, the TRTV-GPKC procedure revealed the best average POR value of 0.010% for the TR problems, while the TRTV-PS composition yielded a slightly larger value (0.015%). Furthermore, VTVM-GPKC, HWC10, and HWC20 produced much larger average POR values for both classes of test

Table 5.2. Average RCP Values for TRTV and VTVM-GPKC (%).

Type	TRTV				VTVM-GPKC
	PS	VA	ADS	GPKC	
LP	1.15	1.57	1.52	1.83	5.54
TR	8.72	10.08	11.04	8.67	132.34
Overall	4.94	5.82	6.28	5.25	68.94

problems.

Recall that the design of the TRTV algorithm was motivated by the desire to achieve robustness. To measure the attained degree of robustness, we computed the standard deviations of the POR values for each class of test problems, and these values are also displayed in Table 5.1. The minimum overall standard deviation (0.040) was produced when the TRTV-ADS composition was implemented, while TRTV-PS and TRTV-VA yielded slightly higher standard deviations (0.043 and 0.045, respectively). Compared with the VTVM-GPKC, HWC10, and HWC20 procedures, the combination TRTV-ADS achieved a significantly lesser standard deviation by the amounts 0.183, 6.751, and 13.620, respectively. To compare the tail-distribution of the solution quality produced by these procedures, we also computed the overall average POR value plus three times the standard deviation for each method. These values for TRTV when implemented along with PS, VA, ADS, and GPKC are 0.160, 0.172, 0.168, and 0.213, respectively, while VTVM-GPKC, HWC10, and HWC20 yielded corresponding values of 0.768, 23.550, and 47.104, respectively. Based on this comparison, TRTV-PS evidently reveals the best robust performance among the implemented algorithms.

To compare CPU times, the average RCP values are presented in Table 5.2. Note that the results for both HWC10 and HWC20 were omitted because these methods could not attain near-optimal values for many test problems as mentioned in Section 4.6. As seen from Table 5.2, TRTV-PS and TRTV-GPKC yielded the best and the second best overall average RCP values of 4.94% and 5.25%, respectively. For the class of LP problems, the PS strategy revealed the best performance (1.15%), while it produced the second best average RCP value of 8.72% for the TR problems. Also, note that GPKC yielded the least RCP value of 8.67% for the TR problems, but it displayed the largest RCP value of 1.83% for the LP problems (among the TRTV variants). Compared with

the VTVM-GPKC procedure, TRTV consistently yielded smaller average RCP values. Note that the overall average RCP values produced by TRTV along with PS, VA, ADS, and GPKC were respectively 64.00%, 63.12%, 62.66%, and 63.60% lesser than those for VTVM-GPKC.

Considering both the POR and RCP values, we recommend that the TRTV along with the PS strategy is the best and the most robust procedure for solving our set of test problems.

5.6 Summary and Conclusions

Observing the non-robustness of the VTVM algorithm and that it may suffer from a crawling phenomenon for some classes of problems, we have developed a new variable target value method, called the *Trust Region Target Value* (TRTV) method, which requires only a few parameters. In this method, a trust region for dual variables is constructed and its center and size are adjusted in a manner that eventually induces a dual optimum to lie at the center of the trust region. A related convergence analysis for this method in concert with a general deflected subgradient scheme as well as with the PS and VA strategies, was also provided.

In our computational study, we implemented the proposed TRTV algorithm in concert with four direction-finding strategies, PS, VA, ADS, and GPKC, and compared their performances with that of VTVM-GPKC procedure. For all four strategies, the TRTV algorithm consistently yielded 0.053-0.071% lesser overall average POR values than that produced by VTVM-GPKC. The best overall average POR value of 0.031% was obtained when TRTV was employed in combination with the PS strategy. In addition to considering the average POR values, we also examined the standard deviations for these POR values in order to compare the relative robustness of the proposed procedures. The TRTV-PS composition yielded the least standard deviation of 0.04, with the standard deviations for TRTV when used with the other strategies also being smaller than that for VTVM-GPKC. Hence, it is evident that TRTV is more robust than the VTVM procedure. As far as CPU times are concerned, the TRTV-PS combination again revealed the best overall average RCP value of 4.94%, which is 64% lesser value than that for VTVM-

GPKC. Therefore, based on our set of test problems, it appears that the TRTV along with the PS strategy is the best and the most robust procedure among those tested.

Chapter 6:
Refinement of Dual Solutions and the Recovery of Primal Solutions

6.1 Introduction

Notice that the algorithms discussed thus far do not possess an appropriate termination criterion since θ would typically be nondifferentiable at points of the optimal solution set, and hence, a zero subgradient might never be realized in practice, even at a true optimum. Furthermore, with a solution of LD alone, we cannot usually derive a primal solution to LP. This can be a handicap when solving LP relaxations of MIPs because in such contexts, a primal solution plays an important role in generating cuts as well as in making branching decisions. We could, however, employ theoretically convergent primal solution recovery schemes as in Shor (1985), Larsson and Liu (1989), and Sherali and Choi (1996), but these require an inordinate number of iterations to be practically effective. With this motivation, we propose in this chapter two dual solution refinement processes along with a primal recovery technique.

In Section 6.2, we explore the use of the dual solution toward optimality, as well as for recovering a primal solution in this process. Computational results are presented in Section 6.3, and we close with some concluding remarks in Section 6.4.

6.2 Dual Refinement and Primal Recovery Technique

Consider the linear program (LR) and its Lagrangian dual formulation (LD) in the form of (1.2) and (1.3), respectively. Suppose that we have a dual solution $\bar{\pi}$ obtained via an NDO algorithm applied to LD. (In particular, we consider the TRTV-PS procedure that revealed the most promising computational results in the previous chapter.) At this stage, we could utilize an outer-linearization method with a trust region to either escape from a jammed non-optimal point, or to refine the solution $\bar{\pi}$ to an optimum π^*, and also, recover a primal solution if $\bar{\pi}$ is close enough to π^*. Observe that we can write the

LD problem in (1.3) as follows.

$$\text{Maximize} \quad v \tag{6.1a}$$

$$\text{subject to} \quad v \le c^T x + \pi^T (b - Ax)\,, \ \forall x \in \{\text{extreme points of } X\}\,. \tag{6.1b}$$

Instead of accommodating the potentially exorbitant set of constraints in (6.1b) (e.g., for all vertices of X), we adopt the usual relaxation of considering the following master program based on some subset of points $x^1, ..., x^j$.

MP: $$\text{Maximize} \quad v \tag{6.2a}$$

$$\text{subject to} \quad v \le c^T x^i + \pi^T (b - Ax^i)\,, \qquad \text{for } i = 1, \ldots, j\,. \tag{6.2b}$$

If a solution to the linear program (6.2) satisfies (6.1b), then we can say that it is optimal to (6.1) as well, and hence, we will have an optimal solution to LD. A primal optimum can also then be recovered by taking a convex combination of x^i, $i = 1, \ldots, j$, using the corresponding dual variables associated with (6.2b) as the weights (see Bazaraa et al., 1993, for example). Let (v_{j+1}, π^{j+1}) be a solution to (6.2) and let x^{j+1} be a corresponding solution to the subproblem (1.3c) for $\pi = \pi^{j+1}$. If $\theta(\pi^{j+1}) \ge v_{j+1}$ (which would then hold as an equality), then (v_{j+1}, π^{j+1}) is optimal to (6.1). Otherwise, we cut off (v_{j+1}, π^{j+1}) by adding the constraint $v \le c^T x^{j+1} + \pi^T (b - Ax^{j+1})$. Based on this outer-linearization method, we consider two trust region schemes to limit the search area, and to accelerate convergence to an optimum given an initial ostensible near-optimal solution $\bar{\pi}$.

In the first method, we use the *Box-step* (BS) method of Marsten et al. (1975), which restricts the feasible region of (6.2) by a box-type trust region, stated in (6.3) below,

$$\bar{\pi} - \Delta e \le \pi \le \bar{\pi} + \Delta e\,, \tag{6.3}$$

where $\Delta > 0$ is the box size parameter, and e is a conformable vector of ones. The

motivation here is that the required piecewise representation of θ in the vicinity of π^*, which is tentatively assumed to be contained within the box (6.3) would be constructed much faster by establishing such a box restriction while reducing the number of cuts generated and avoiding excessive oscillations in the solution space.

Let us denote $(6.1)_B$ and $(6.2)_B$ (or MP_B) to be the respective problems (6.1) and (6.2) in which the dual space is further restricted by the box constraints (6.3), and suppose that $(\tilde{z}, \tilde{\pi})$ is an optimal solution obtained for $(6.1)_B$ via the outer-linearization method. If $\tilde{\pi}$ is an interior point of the box, then it is an optimal solution to (6.1), and hence, $\tilde{\pi}$ is also an optimal solution to LD. Otherwise, the box is re-centered at the incumbent point $\tilde{\pi}$, and the solution of MP_B is continued using the available set of cuts of type (6.2b). Furthermore, we modify Δ by commencing with a value of $\Delta = \max\{0.05\mu_{final}, 0.001\}$, where μ_{final} is the resulting trust region parameter value at the end of the TRTV-PS procedure. Thereafter, whenever a new box is established, Δ is inflated by a factor $\gamma_2 \geq 1$ to speed up the inclusion of an optimal solution until a threshold value $\overline{\Delta}$ is reached. By repeating this procedure until an optimal solution $(\tilde{\pi})$ is contained in the interior of the current box, we can derive a set of primal and dual optimal solutions.

In our second proposed procedure, called the *Box Trust Region* (BTR) method, instead of retaining the same trust region throughout each outer-linearization loop, we continually relocate the trust region and adjust its size according to the progress with respect to the incumbent objective value. As in the boxstep method, if we have a solution (v_{j+1}, π^{j+1}) to MP_B that satisfies the optimality condition $\theta(\pi^{j+1}) \geq v_{j+1}$, then we declare π^{j+1} to be optimal to (6.1) (or LD) if it lies in the interior of the current box, or else, we expand this box, re-centering it at π^{j+1}, and continue the algorithmic process. On the other hand, suppose that (v_{j+1}, π^{j+1}) does not satisfy the optimality condition for $(6.1)_B$. Consider the incumbent solution defined by $\overline{\pi}$, where $\theta(\overline{\pi}) = \max\{\theta(\pi^i), i = 1,..., j+1\}$ and where π^1 is the initial near-optimal solution. If $\theta(\pi^{j+1}) < \theta(\overline{\pi})$, then Δ is reduced since we suspect that a too large trust region induced a larger step-length. On the other hand, suppose that the incumbent solution is π^{j+1} itself. If π^{j+1} is in the interior of the

current box, the box size Δ is reduced; otherwise, Δ is increased to encourage further exploration. In either case, the box is re-centered at the incumbent solution at this step itself.

Two outer-linearization refinement methods in conjunction with trust region strategies are summarized below.

Refinement Using the Boxstep Method

Step 1: Given a solution $\bar{\pi}$ from an NDO algorithm, let x^1 be a solution to (1.3c) corresponding to $\pi^1 = \bar{\pi}$. Select an optimality tolerance $\varepsilon_{opt} > 0$, a box size parameter $\Delta_1 > 0$ (for example, $\Delta_1 \equiv \max\{0.05\mu_{final}, 0.001\}$ as above), and a threshold box size parameter $\bar{\Delta} > \Delta_1$. Put the iteration counter $j = 1$, the box update iteration counter $l = 1$, and select an inflation parameter $\gamma_2 \geq 1$.

Step 2: Solve the master program (6.2) along with the following box constraint by calling an LP solver:

$$\bar{\pi} - \Delta_l e \leq \pi \leq \bar{\pi} + \Delta_l e .$$

Let (v_{j+1}, π^{j+1}) be the solution obtained with duals σ_i, $i = 1, \ldots, j$, associated with (6.2b). Evaluate $\theta(\pi^{j+1})$, and let x^{j+1} be the corresponding solution to the subproblem (1.3c). If $\theta(\pi^{j+1}) \geq v_{j+1} - \varepsilon_{opt}$, then proceed to Step 3. Otherwise, increment $j \leftarrow j+1$ and repeat Step 2.

Step 3: If $\bar{\pi} - \Delta_l e < \pi^{j+1} < \bar{\pi} + \Delta_l e$, stop with π^{j+1} as the prescribed dual solution, and compute the primal (near-) optimum as $\sum_{i=1}^{j} \sigma_i x^i$. Otherwise, increase the box size to $\Delta_{l+1} = \gamma_2 \Delta_l$ and replace $\bar{\pi} \leftarrow \pi^{j+1}$. If $\Delta_{l+1} > \bar{\Delta}$, return to the NDO algorithm with $\bar{\pi}$. Otherwise, increment $l \leftarrow l+1$, and return to Step 2.

Refinement Using a Box Trust Region

Step 1: Given a solution $\bar{\pi}$ from an NDO algorithm, let x^1 be a solution to (1.3b) corresponding to $\pi^1 = \bar{\pi}$. Select an optimality tolerance $\varepsilon_{opt} > 0$, a box size parameter $\Delta_1 > 0$ (for example, $\Delta_1 \equiv \max\{0.05\mu_{final}, 0.001\}$ as above), and a threshold box size parameter $\bar{\Delta} > \Delta_1$. Put the iteration counter $j = 1$, the box update iteration counter $l = 1$, and select inflation and deflation parameters $\gamma_2 \geq 1$ and $\gamma_3 \in (0,1]$, respectively.

Step 2: Solve the master program (6.2) along with the following box constraint by calling an LP solver:

$$\bar{\pi} - \Delta_l e \leq \pi \leq \bar{\pi} + \Delta_l e .$$

Let (v_{j+1}, π^{j+1}) be the optimal solution obtained for (6.2), with duals σ_i, $i = 1, \ldots, j$, associated with (6.2b). Evaluate $\theta(\pi^{j+1})$, and let x^{j+1} be the corresponding solution to this subproblem.

Step 3: If $\theta(\pi^{j+1}) \geq v_{j+1} - \varepsilon_{opt}$, then go to Step 5. Else, if $\theta(\pi^{j+1}) > \theta(\bar{\pi})$, update the incumbent solution $\bar{\pi} \leftarrow \pi^j$, and proceed to Step 4. Otherwise, update $\Delta_{l+1} = \gamma_3 \Delta_l$, increment $j \leftarrow j+1$, $l \leftarrow l+1$, and return to Step 2.

Step 4: If $\bar{\pi} - \Delta_l e < \pi^{j+1} < \bar{\pi} + \Delta_l e$, update $\Delta_{l+1} = \gamma_3 \Delta_l$; else, update $\Delta_{l+1} = \gamma_2 \Delta_l$. Increment $j \leftarrow j+1$, $l \leftarrow l+1$, and return to Step 2.

Step 5: If $\bar{\pi} - \Delta_l e < \pi^{j+1} < \bar{\pi} + \Delta_l e$, stop with π^{j+1} as the prescribed dual solution, and compute the primal (near-) optimum as $\sum_{i=1}^{j} \sigma_i x^i$. Otherwise, put $\bar{\pi} \leftarrow \pi^{j+1}$, $\Delta_{l+1} = \gamma_2 \Delta_l$. If $\Delta_{l+1} > \bar{\Delta}$, return to the NDO algorithm with $\bar{\pi}$. Otherwise, increment $l \leftarrow l+1$, and return to Step 2.

Remark 6.1: If the number of constraints for the master program exceeds a certain threshold value when we add a new constraint, instead of keeping all the existing cuts, we can delete the nonbinding cuts from the previous iteration for the sake of computational effectiveness. We shall explore this feature in our empirical study.

Table 6.1. Average Percentage POR Ratio for Dual Refinements (%).

Type	BS	BTR	BS-D	BTR-D
LP	60.9	39.5	60.2	37.0
TR	97.9	97.8	97.6	97.4
Overall	79.4	68.6	78.9	67.2

6.3 Computational Study

We implemented the two dual solution refinement and primal recovery methods: Box-step (BS) and Box Trust Region (BTR), in concert with the solutions produced by the TRTV-PS procedure, which revealed the best performance in the computational experiment in Chapter 5. Note that, in order to recover a primal solution, the memory requirement for storing solutions to subproblems increases proportionately with the number of cuts. Hence, for the purpose of compactness and efficiency, we terminated the refinement procedure when the number of constraints in the master program reached 50. The box-size parameter was initialized as $\Delta_1 \equiv \max\{0.05\mu_{final}, 0.001\}$, where μ_{final} is the resulting trust region parameter value at the end of the TRTV-PS procedure. Based on our preliminary experiment with several parameter values, we selected $\varepsilon_{opt} = 1$, $\overline{\Delta} = \infty$, $\gamma_2 = 2$, and $\gamma_3 = 0.99$. As in Remark 6.1, we also experimented with deleting nonbinding constraints when the number of cuts exceeded 30. (Let us denote these procedures by BS-D and BTR-D for the Box-step and the Box Trust Region methods, respectively.)

To examine the efficacy of the proposed dual refinement methods, we present in Table 6.1 the values of the statistic: *Percentage POR Ratio*, which is defined as {[POR value produced after applying the refinement method] ÷ [POR value produced by the stand-alone TRTV-PS procedure]} ×100%. Hence a lower percentage value indicates a greater further reduction in the optimality gap with respect to that for the TRTV-PS method. From Table 6.1, BTR-D yielded the best overall average Percentage POR Ratio of 67.2%, while BTR produced the second best value of 68.6%. For the class of LP problems, the proposed refinements methods further reduced the (near-zero) POR values produced by TRTV-PS by 39.1-60.9%. However, the corresponding average decrements

Table 6.2 Comparison of Average Degree of Primal Infeasibility.

Average for	Proposed Primal Recovery Scheme				Ergodic Primal Recovery Scheme		
	BS	BTR	BS-D	BTR-D	LPS	SHOR	SC
LP	2.65	0.54	2.63	0.67	12.43	4.30	2.75
TR	5.60	4.35	5.38	3.86	8.61	2.34	28.40
All Problems	4.13	2.45	4.00	2.26	10.52	3.32	15.58

were only 2.1-2.6% for the TR problems. Also, note that deleting nonbinding constraints yielded slightly smaller Percentage POR Ratio values. In this case, because we delete nonbinding constraints, these procedures usually perform more iterations than do BS and BTR before reaching the prescribed maximum number of cuts. Hence, BS-D and BTR-D tend to produce better incumbent solutions. Furthermore, BTR(-D) consistently outperformed BS(-D). This can be explained by noting that since BTR continually adjusts the center of the trust region to the incumbent solution as soon as it is revised, the box in the BTR method is more likely to include an optimal solution earlier than in the BS approach.

Finally, in order to examine the performance with respect to recovering primal solutions, we report on the average degree of primal infeasibility (ρ, see Section 4.6) attained, along with the corresponding ρ values for three ergodic recovery schemes in Table 6.2. Among the studied schemes, the best overall average ρ value of 2.26 was produced by the BTR-D method, while BTR was a close second-best ($\rho = 2.45$). Compared with the BS method, BTR revealed a consistently better performance, and again, this can be explained by the trust region update scheme. Since BTR continually adjusts the trust region so that the incumbent solution can be located at its center, a solution of the master program is likely to be an interior point of the trust region. Hence, the corresponding cutting planes tend to better define a dual optimum than those in the BS method. Compared with the ergodic primal recovery schemes, BTR achieved a smaller degree of primal infeasibility at an average than the value $\rho = 3.32$ that was produced by SHOR. Considering that SHOR utilized the (typically unknown) optimal value for its dual optimization procedure, and BTR was terminated after only 50

constraints were generated, these results exhibit a good promise for the proposed primal recovery scheme.

6.4 Summary and Conclusions

In this chapter, we have explored two dual refinement and primal recovery procedures, BS and BTR. Assuming that a near-optimal solution is at hand, these methods utilize the outer linearization (or cutting-plane) method within a small trust region in the hope of avoiding the generation of a potentially excessive set of constraints. The trust region is updated as prescribed in each method so that a dual optimum can eventually be included in its interior.

In concert with the solutions produced by the TRTV-PS procedure, we implemented BS and BTR until the number of constraints reached 50. We also experimented with deleting nonbinding constraints whenever the number of cuts exceeds 30. Based on our computational results, the BTR method, while deleting nonbinding constraints (BTR-D), displayed the best performance with respect to both dual refinement and primal recovery. We observed that BTR-D further decreased the near-zero POR values produced by TRTV-PS by 38.2% on the average. Furthermore, compared with several ergodic primal recovery schemes, the BTR-D procedure consistently yielded a smaller degree of primal infeasibility. Noting that it was implemented until only 50 constraints were generated, these results portend a good promise for the BTR-D method.

Chapter 7:
Research Contributions and
Recommendations for Future Research

Research Contributions

When solving large-scale, ill-conditioned linear programming relaxations of mixed-integer programs in the context of branch-and-bound/cut methods, it is well known that simplex-based and interior point algorithms can stall and fail to produce solutions with adequate speed and accuracy. Instead, Lagrangian relaxation approaches along with a primal recovery scheme can be used to derive quick bounds as well as for selecting branching variables. However, the Lagrangian dual function of a linear program is typically nondifferentiable. Although the pure subgradient algorithm is a popularly used nondifferentiable optimization (NDO) technique due to its simplicity, it suffers from a very slow convergence rate. Also, to be implemented in practice, it requires the prescription of a suitable upper bound on the optimal objective value for the purpose of computing step-lengths. Furthermore, other approaches, such as the bundle method and space dilation techniques, are not suitable for solving large-sized problems because of their excessive computational and storage requirements. Our research effort has therefore focused on developing efficient nondifferentiable optimization techniques for solving Lagrangian duals of large-sized linear programs, as well as on deriving primal solutions via practical, effective primal recovery schemes.

In Chapter 2, we embedded the Volume Algorithm (VA) of Barahona and Anbil (2000) within the Variable Target Value Method (VTVM) framework of Sherali et al. (2000) as a deflected subgradient strategy, and provided a convergence analysis under the condition that the geometric moving average parameter α and the step-length factor β satisfy $0 < \beta \leq \alpha \leq 1$. Since a target value is provided via VTVM, our adaptation affords a generic implementation of the VA strategy without the need for devising specialized heuristics for computing upper bounds, tailored for each specific class of problems. We

also presented an example that demonstrates the primal non-convergence of the original Volume Algorithm. To test the computational performance of embedding the proposed adaptation VA within VTVM, we implemented this method along with other competing subgradient deflection strategies within the VTVM framework. Based on our computational results, the VA strategy revealed a promising performance together with the Polyak-Kelley cutting-plane (PKC) and the average direction (ADS) strategies, designed by Sherali et al. (2001), and Sherali and Ulular (1989), respectively.

Next, in Chapter 3, we considered two variable target value methods, the Level Algorithm and VTVM, which do not require any prior information on the optimal objective value. Noting that PKC revealed a promising performance from Chapter 1, we first designed a generalized version GPKC, in which the cuts employ an estimate of the optimal objective value based on a lag of some $\bar{\delta}' \geq 0$ iterations and also incorporate sequential projections involving some $\bar{\delta}'' \geq 1$ previous cuts in addition to that for the current iteration. We also provided convergence analyses for the Level Algorithm when used in concert with VA, ADS, and PKC, and established the convergence for VTVM when used in conjunction with GPKC. Furthermore, observing some potential weaknesses in the target value updating schemes for these two variable target value frameworks, we proposed modifications to remedy these shortcomings. In addition, a projected-quadratic line-search was incorporated as a further enhancement. Based on our extensive computational study, the modified VTVM in concert with GPKC having parameter values $\bar{\delta}' = 0$ and $\bar{\delta}'' = 4$ produced the best quality solutions among all implemented procedures. We also observed that the proposed modifications in updating the target values and the projected quadratic-fit line-search played a beneficial role. Between the two variable target value frameworks, VTVM consistently outperformed the Level Algorithm. Moreover, a CPU time comparison demonstrated that the modified VTVM in conjunction with GPKC consumed only 29.74% of the computational effort required by the commercial LP solver, CPLEX 8.1, in order to attain the same quality solution.

In Chapter 4, we developed two novel approaches, a perturbation technique (PT) and a barrier-Lagrangian dual reformulation method (BLR), which either attempt to circumvent or obviate the nondifferentiability of the Lagrangian dual function so that conventional differentiable optimization methods could be directly employed to derive good initial solutions and target values for NDO algorithms. Whenever encountering a nondifferentiable point, the PT procedure finds a perturbed differentiable solution from which it continues the search. On the other hand, the BLR method provides a (twice) differentiable Lagrangian dual function via a barrier problem, thereby assuring an ascent scheme. Since a differentiable point is available at any iteration, these methods afford employing conventional deflected gradient schemes including these of Hestenes and Stiefel (1952, HS), Polak and Ribiere (1969, PR), Fletcher and Reeves (1964, FR), Sherali and Ulular (1990, SU), Sherali and Ulular (1989, ADS), and Shanno (1978, BFGS). Using these constructs, we proposed a two-phase algorithmic process. In the first phase (Phase I) of this method, PT or BLR are utilized to discover a relatively good solution for offering a warm-start to Phase II. Then, the modified VTVM technique along with the GPKC direction and step-length strategy having parameter values $\bar{\delta}' = 0$ and $\bar{\delta}'' = 4$ (VTVM-GPKC) is implemented in Phase II. Based on our computational results, both PT and BLR of Phase I helped significantly improve the algorithmic performance as compared with implementing VTVM-GPKC alone, even though the latter was run for twice as many iterations as the two-phase procedures. Among all combinations, PT-BFGS, along with a target value initialization method using a projected estimation for Phase II, yielded the best overall quality solutions, producing an average POR value of 0.028%, which measures the relative final optimality gap to the initial optimality gap. Also, the CPU time consumed by this combination in order to attain a prescribed near-optimal solution at an average was only 5.56% of that required by CPLEX 8.1.

Observing the non-robustness of the VTVM algorithm and that it may suffer from a crawling phenomenon for some classes of problems, we develop in Chapter 5 a new variable target value method, called the *Trust Region Target Value* (TRTV) method, which requires the management of only a few parameters. In this method, a trust region is constructed for dual variables and its center and size are adjusted in a manner that

eventually induces a dual optimum to lie at the center of the trust region. A related convergence analysis for this method in concert with a general deflected subgradient scheme as well as with the PS and VA strategies, has also been provided. To examine its computational performance, we implemented the TRTV algorithm in concert with four direction-finding strategies, PS, VA, ADS, and GPKC. Compared with VTVM-GPKC, the TRTV algorithm consistently yielded 0.053-0.071% lesser overall average POR values for all four strategies. The best overall average POR value of 0.031% was obtained when TRTV was employed in concert with the PS strategy. As a measurement of robustness, we also examined the standard deviations for these POR values. In consequence, the TRTV-PS composition yielded the least standard deviation of 0.04, with the standard deviations for TRTV when used with the other strategies also being smaller than that for VTVM-GPKC. Hence, it is evident that TRTV is more robust than the VTVM procedure. Furthermore, the TRTV-PS combination revealed the best CPU time performance as well by consuming only 4.94% of the effort required by CPLEX 8.1 to produce comparable solutions.

Finally, in Chapter 6, we explored two dual refinement and primal recovery procedures, BS and BTR. Assuming that a near-optimal solution is available via a suitable nondifferentiable optimization method, we utilized the outer linearization (or cutting-plane) method within a small trust region in the hope that we could avoid generating a potentially excessive set of constraints. The trust region is updated as necessary in these methods so that a dual optimum is eventually included in its interior. We implemented the BS and BTR methods in combination with the TRTV-PS procedure that revealed the best performance in Chapter 5. We also experimented with deleting nonbinding constraints whenever the number of cuts exceeded a certain threshold value. Based on our computational results, BTR along with the deletion of nonbinding constraints (BTR-D) displayed the best performance with respect to both dual refinement and primal recovery. We observed that BTR-D further decreased the POR values produced by TRTV-PS by 38.2% on average. Furthermore, compared with several ergodic primal recovery schemes, the BTR-D procedure produced primal solutions that had a significantly higher degree of primal feasibility. Noting that this scheme was

terminated after only 50 active constraints had accumulated, these results portend a good promise for the proposed BTR-D method.

Recommendations for Future Research

First, noting that Phase I of the two-phase procedure proposed in Chapter 4 helped enhance the performance of the NDO algorithms, we recommend combining the PT or BLR with the TRTV-PS procedure that revealed the best performance in Chapter 5. Furthermore, at the end of this combined procedure, it is worth to implement the BTR-D method designed in Chapter 6, since we observed that this technique further enhanced the solution quality in our set of test problems. In consequence, the three-phase method that consists of PT or BLR in Phase I, TRTV-PS in Phase II, and BTR-D in Phase III, is recommended for a starting point for future research. Furthermore, when only a part of the primal constraints are dualized in the Lagrangian dual formulation, we need to establish a suitable relationship between the solution to Phase I and the starting solution to Phase II.

Next, note that the BLR phase significantly enhanced the ergodic primal recovery performance and the BTR-D method yielded a better primal solution when compared with existing ergodic schemes. Hence, we recommend further research on combining ergodic and trust region approaches. For this to be successful, we need to first design an effective ergodic primal recovery scheme for the TRTV-PS procedure. In addition, a suitable adoption of the resulting ergodic primal solution into the BTR-D method must be studied.

In our research, we have observed that we could potentially reduce computational effort to solve discrete problems by using the proposed methods in the context of Lagrangian relaxation approaches, or by optimizing suitable Lagrangian dual formulations, instead of directly solving the linear programming relaxations using an LP solver in the context of LP-based branch-and-bound/cut algorithms. Therefore, an actual study of this potential benefit is our final recommendation for future study.

Appendix I:

Twice-differentiability of the Lagrangian Dual Function for the Barrier Problem

Using (4.14a,b), the derivative of $\bar{x}_j$, viewed as a function $\bar{x}_j(\bar{c}_j)$ of $\bar{c}_j$, at the point $\bar{c}_j = 0$, yields using L'Hospital's Rule,

$$\left.\frac{d\bar{x}_j(\bar{c}_j)}{d\bar{c}_j}\right|_{\bar{c}_j=0} = \lim_{\Delta\bar{c}_j\to 0}\frac{\bar{x}_j(\Delta\bar{c}_j)-\bar{x}_j(0)}{\Delta\bar{c}_j}$$

$$= \lim_{\Delta\bar{c}_j\to 0}\frac{\dfrac{\Delta\bar{c}_j(l_j+u_j)+2\mu-\sqrt{\Delta\bar{c}_j^{\,2}(u_j-l_j)^2+4\mu^2}}{2\Delta\bar{c}_j}-\dfrac{l_j+u_j}{2}}{\Delta\bar{c}_j}$$

$$= \lim_{\Delta\bar{c}_j\to 0}\frac{2\mu-\sqrt{\Delta\bar{c}_j^{\,2}(u_j-l_j)^2+4\mu^2}}{2\Delta\bar{c}_j^{\,2}}$$

$$= \lim_{\Delta\bar{c}_j\to 0}\frac{-\Delta\bar{c}_j(u_j-l_j)^2}{4\Delta\bar{c}_j\sqrt{\Delta\bar{c}_j^{\,2}(u_j-l_j)^2+4\mu^2}}$$

$$= \lim_{\Delta\bar{c}_j\to 0}\frac{-(u_j-l_j)^2}{4\sqrt{\Delta\bar{c}_j^{\,2}(u_j-l_j)^2+4\mu^2}} = \frac{-(u_j-l_j)^2}{8\mu}. \qquad \text{(A1)}$$

On the other hand, we have that

$$\left.\frac{d\bar{x}_j(\bar{c}_j)}{d\bar{c}_j}\right|_{\bar{c}_j\neq 0} = \frac{\left[(l_j+u_j)-\frac{1}{2}\left(2\bar{c}_j(u_j-l_j)^2\right)\left(\bar{c}_j^{\,2}(u_j-l_j)^2+4\mu^2\right)^{-1/2}\right](2\bar{c}_j)}{(2\bar{c}_j)^2}$$

$$-\frac{2\left[\bar{c}_j(l_j+u_j)+2\mu-\left(\bar{c}_j^{\,2}(u_j-l_j)^2+4\mu^2\right)^{1/2}\right]}{(2\bar{c}_j)^2}$$

$$= \frac{2\left(\bar{c}_j^{\,2}(u_j-l_j)^2+4\mu^2\right)^{1/2}-2\bar{c}_j^{\,2}(u_j-l_j)^2\left(\bar{c}_j^{\,2}(u_j-l_j)^2+4\mu^2\right)^{-1/2}-4\mu}{4\bar{c}_j^{\,2}}.$$

Hence, using L'Hospital's Rule we have

$$\lim_{\bar{c}_j \to 0} \frac{d\bar{x}_j(\bar{c}_j)}{d\bar{c}_j} = \lim_{\bar{c}_j \to 0} \left[\frac{2\bar{c}_j (u_j - l_j)^2 \left(\bar{c}_j{}^2 (u_j - l_j)^2 + 4\mu^2 \right)^{-1/2}}{8\bar{c}_j} \right.$$

$$- \frac{4\bar{c}_j (u_j - l_j)^2 \left(\bar{c}_j{}^2 (u_j - l_j)^2 + 4\mu^2 \right)^{-1/2}}{8\bar{c}_j}$$

$$\left. + \frac{\bar{c}_j{}^2 (u_j - l_j)^2 \left(2\bar{c}_j (u_j - l_j)^2 \right) \left(\bar{c}_j{}^2 (u_j - l_j)^2 + 4\mu^2 \right)^{-3/2}}{8\bar{c}_j} \right]$$

$$= \lim_{\bar{c}_j \to 0} \left[\frac{-2(u_j - l_j)^2 \left(\bar{c}_j{}^2 (u_j - l_j)^2 + 4\mu^2 \right)^{-1/2}}{8} \right.$$

$$\left. + \frac{2\bar{c}_j{}^2 (u_j - l_j)^4 \left(\bar{c}_j{}^2 (u_j - l_j)^2 + 4\mu^2 \right)^{-3/2}}{8} \right]$$

$$= -\frac{(u_j - l_j)^2}{8\mu}. \qquad \text{(A2)}$$

From (A1) and (A2), we see that $\left. \frac{d\bar{x}_j(\bar{c}_j)}{d\bar{c}_j} \right|_{\bar{c}_j = 0} = \lim_{\bar{c}_j \to 0} \frac{d\bar{x}_j}{d\bar{c}_j} = \frac{-(u_j - l_j)^2}{8\mu}$.

References

[1] Adams, W. P. and H. D. Sherali, "Mixed-Integer Bilinear-Programming Problems," *Mathematical Programming*, 59(3), 279-305, 1993.

[2] Bahiense, L., N. Maculan, and C. Sagastizábal, "The Volume Algorithm Revisited: Relation with Bundle Methods," *Mathematical Programming*, 94(1), 41-69, 2002.

[3] Barahona, F. and R. Anbil, "The Volume Algorithm: Producing Primal Solutions with a Subgradient Method," *Mathematical Programming*, 87(3), 385-399, 2000.

[4] Bazaraa, M. S. and H. D. Sherali, "On the Choice of Step Size in Subgradient Optimization," *European Journal of Operational Research*, 7(4), 380-388, 1981.

[5] Bazaraa, M. S., H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd ed., John Wiley & Sons: New York, 1993.

[6] Bazaraa, M. S., J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, 2nd ed., John Wiley & Sons: New York, 1990.

[7] Beasley, J. E., "OR-Library: Distributing Test Problems by Electronic Mail," *Journal of the Operational Research Society*, 41(11), 1069-1072, 1990.

[8] Brännlund, U., *On Relaxation Methods for Nonsmooth Convex Optimization*, Ph.D Dissertation, Department of Mathematics, Royal Institute of Technology, Stockholm, Sweden, 1993.

[9] Brännlund, U., K. C. Kiwiel, and P. O. Lindberg, "A Descent Proximal Level Bundle Method for Convex Nondifferentiable Optimization," *Operations Research Letters*, 17(3), 121-126, 1995.

[10] Camerini, P. M., L. Fratta, and F. Maffioli, "On Improving Relaxation Methods by Modified Gradient Techniques," *Mathematical Programming Study*, 3, 26-34, 1975.

[11] Caprara, A., M. Fischetti, and P. Toth, "A Heuristic Method for the Set Covering Problem," *Operations Research*, 47(5), 730-743, 1999.

[12] Fisher, M.L., "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science*, 27 (1), 1-18, 1981.

[13] Fletcher, R. and C.M. Reeves, "Function Minimization by Conjugate Gradients," *Computer Journal*, 7 (2), 149-154, 1964.

[14] Fumero, F., "A Modified Subgradient Algorithm for Lagrangean Relaxation," *Computers & Operations Research*, 28(1), 33-52, 2001.

[15] Goffin, J. L. and K. C. Kiwiel, "Convergence of a Simple Subgradient Level Method," *Mathematical Programming*, 85(1), 207-211, 1999.

[16] Goffin, J. L. and J. P. Vial, "Convex Nondifferentiable Optimization: a Survey Focused on the Analytic Center Cutting Plane Method," *Optimization Methods and Software*, 17, 805-867, 2002.

[17] Held, M., P. Wolfe, and H. Crowder, "Validation of Subgradient Optimization," *Mathematical Programming*, 6(1), 62-88, 1974.

[18] Hestenes, M. R. and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards, Section B*, 48, 409-436, 1952.

[19] Hiriart-Urruty, J. B. and C. Lemarechal, *Convex Analysis and Minimization Algorithms*, Springer-Verlag: New York, N.Y., 1993.

[20] Hoffman, K. L. and M. Padberg, "Solving Airline Crew Scheduling Problems by Branch-and-Cut", Management Science, 39(6), 657-82, 1993.

[21] Kim, S., H. Ahn, and S. Cho, "Variable Target Value Subgradient Method," *Mathematical Programming*, 49(3), 359-369, 1991.

[22] Kiwiel, K. C., "A Tilted Cutting Plane Proximal Bundle Method for Convex Nondifferentiable Optimization," *Operations Research Letters*, 10(2), 75-81, 1991.

[23] Konnov, I. V., "On Convergence Properties of a Subgradient Method," *Optimization Methods and Software*, 18(1), 53-62, 2003.

[24] Larsson, T. and Z. Liu, *A Primal Convergence Result for Dual Subgradient Optimization with Application to Multi-commodity Network Flows*, Research Report, Dept. of Math., Linkoping Institute of Technology, S-581 83 Linkoping, Sweden, 1989.

[25] Larsson, T., M. Patriksson, and A. B. Stromberg, "Ergodic, Primal Convergence in Dual Subgradient Schemes for Convex Programming," *Mathematical Programming*, 86(2), 283-312, 1999.

[26] Lemarechal, C., "Bundle Method in Nonsmooth Optimization," in Nonsmooth Optimization: *Proceedings of IIASA Workshop* held at Laxenburg, Austria, C. Lemarechal and R. Mifflin (Eds.), 3, 79-109, 1977.

[27] Makela, M. M., "Survey of Bundle Methods for Nonsmooth Optimization," *Optimization Methods and Software*, 17(1), 1-29, 2002.

[28] Mifflin, R., "An Algorithm for Constrained Optimization with Semismooth Functions," *Mathematics of Operations Research*, 2(2), 191-207, 1977.

[29] Marsten, R. E., W. W. Hogan, and J. W. Blankenship, "The Boxstep Method for Large-Scale Optimization," *Operations Research*, 23 (3) 389-405, 1975.

[30] Polak E. and G. Ribiere, "Note on the Convergence of Methods of Conjugate Directions," *Revue Francaise D'Informatique et de Recherche Operationelle*, 3(16) 35-43, 1969.

[31] Polyak, B. T., "A General Method of Solving Extremum Problems," *Soviet Mathematics*, 8 (3), 593-597, 1967.

[32] Polyak, B. T., "Minimization of Unsmooth Functionals," *U.S.S.R. Computational Mathematics and Mathematical Physics*, 9(3), 14-29, 1969.

[33] Powell, M. J. D., "Restart Procedures for the Conjugate Gradient Method," *Mathematical Programming*, 12 (2), 241-254, 1977.

[34] Rockafellar, R. T., *Convex Analysis*, Princeton University Press, Princeton, New Jersey, 1970.

[35] Rosen, J. B. and S. Suzuki, "Construction of Nonlinear Programming Test Problems," *Communication of ACM*, 8(2), 113, 1965.

[36] Sen, S. and H. D. Sherali, "A Class of Convergent Primal-Dual Subgradient Algorithms for Decomposable Convex Programs," *Mathematical Programming*, 35(3), 279-297, 1986.

[37] Shanno, D. F., "Conjugate Gradient Methods with Inexact Searches," *Mathematics of Operations Research*, 3 (3), 244-256, 1978.

[38] Sherali, H. D. and G. Choi, "Recovery of Primal Solutions When Using Subgradient Optimization Methods to Solve Lagrangian Duals of Linear Programs," *Operations Research Letters*, 19(3), 105-113, 1996.

[39] Sherali, H. D., G. Choi, and Z. Ansari, "Limited Memory Space Dilation and Reduction Algorithms," *Computational Optimization and Applications*, 19(1), 55-77, 2001.

[40] Sherali, H. D., G. Choi, and C. H. Tuncbilek, "A Variable Target Value Method for Nondifferentiable Optimization," *Operations Research Letters*, 26(1), 1-8, 2000.

[41] Sherali, H. D. and D. C. Myers, "Dual Formulations and Subgradient Optimization Strategies for Linear Programming Relaxations of Mixed-Integer Programs," *Discrete Applied Mathematics*, 20(1), 51-68, 1988.

[42] Sherali, H. D. and C. H. Tuncbilek, "New Reformulation Linearization/ Convexification Relaxations for Univariate and Multivariate Polynomial Programming Problems," *Operations Research Letters*, 21(1), 1-9, 1997.

[43] Sherali, H. D. and O. Ulular, "A Primal-Dual Conjugate Subgradient Algorithm for Specially Structured Linear and Convex Programming Problems," *Applied Mathematics and Optimization*, 20(2), 193-221, 1989.

[44] Sherali, H. D. and O. Ulular, "Conjugate Gradient Methods Using Quasi-Newton Updates with Inexact Line Searches," *Journal of Mathematical Analysis and Applications*, 150(2), 359-377, 1990.

[45] Shor, N. Z., "Utilization of the Operation of Space Dilatation in the Minimization of Convex Functions," *Kibernetika*, 6(1), 6-12, 1970.

[46] Shor, N. Z., "Convergence Rate of the Gradient Descent Method with Dilatation of the Space," *Kibernetika*, 6(2), 80-85, 1970.

[47] Shor, N. Z., *Minimization Methods for Non-Differentiable Functions*, Springer-Verlag, 1985.

[48] Shor, N. Z. and L. P. Shabashova, "Solution of Minimax Problems by the Method of Generalized Gradient Descent with Dilatation of the Space," *Kibernetika*, 8(1), 82-88, 1972.

[49] Shor, N. Z. and N. G. Zhurbenko, "A Minimization Method Using the Operation of Extension of the Space in the Direction of the Difference of Two Successive Gradients," *Kibernetika*, 7(2), 51-59, 1971.

Vita

Churlzu Lim finished his Ph.D. degree in the Grado Department of Industrial and Systems Engineering at Virginia Polytechnic Institute and State University in July 2004. During his Ph.D. studies, Churlzu received the Grado Fellowship, recognizing his academic merit and research promise. His B.S. and M.S. degrees were completed at Korea Advanced Institute of Science and Technology in 1995 and 1997, respectively.

Churlzu was born in 1971 and married Hye-Young Kim in 2000. He has a ten-month-old daughter, Harin, as of July 2004.